République Algérienne Démocratique et Populaire.
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique.

Université Saad Dahlab, Blida
USDB.

Faculté des sciences.
Département informatique .

Mémoire pour l'obtention
d'un diplôme d'ingénieur d'état en informatique.
Option : SI

Sujet :

Implémentation d'un agent superviseur qui aide à la modélisation des procédés logiciels

Présenté par :Méziani Farid
Chaa youcef

Promoteur : Aoussat Fadila

Soutenue le: 28/09/2005, devant le jury composé de :

Mlle Boustia	**Président**
Mlle Fareh	**Examinateur**
Mr Boukhlef	**Examinateur**

-2004/2005-

Remerciements

Tout d'abord nous remercions le bon **Dieu** pour nous avoir guider vers le bon chemin de savoir, pour nous avoir donner du courage et de la volonté afin de pouvoir réaliser ce mémoire.

Nous exprimons nos sincères remerciements à nos parents pour tout ce qu'ils ont pu nous apporter.

Nous tenant à remercier les membres du jury pour avoir eu l'obligeance d'accepter et d'apprécier notre travail.

Nous tenons également à exprimer notre gratitude à notre promotrice Mm. Aoussat pour sa préoccupation.

A tout les enseignants de la faculté des sciences exacts et surtout les enseignants du département informatique.

En fin, tout ceux qui nous ont aidé de pré ou de loin à la réalisation de ce mémoire, en particulier D.Omar, D.Zaimi, B.Salah Amine...

Dédicaces

Je dédie mon travail à:

Mon cher papa et ma chère maman à qui je souhaite une longue vie.

Mes sœurs et mon frère.

Toute ma famille.

Tous mes amis sans exception.

Et surtout à mon binôme **Youcef.**

Tout ceux qui m'on aider de près ou de loin.

Je dédie mon travail à:

Mon cher papa et ma chère maman à qui je souhaite une longue vie.

Mes sœurs et mes frères.

Toute ma famille.

Tout mes amis sans exception.

Et surtout à mon binôme **Farid.**

Tout ceux qui m'on aider de près ou de loin.

Résumé

La modélisation des procédés logiciels doit permettre la représentation de toute les caractéristiques et spécificités du développement logiciels, le modèle de procédé logiciel idéale doit ainsi être suffisamment flexible pour permettre la prise en charge des modifications pendant l'exécution, sans pour autant prendre en vue les objectifs initiaux ; il doit permettre une part d'autonomie locale tout en gardant la cohérence globale du développement ; il doit en autre supporter l'hétérogénéité des outils et langage de communication en substance notre modèle de procédé doit être intelligent.

Dans cette thèse nous prenons la nouvelle approche pour la modélisation et l'exécution des procédés logiciels à base d'agents intelligents et nous essayons de réaliser et implémenter la partie supérieure du système multi agent hiérarchique (AGENT SUPERVISEUR).

La première partie est un état d'art relative à l'utilisation des agents intelligents dans l'ingénierie logiciel, et la présentation du système multi agent hiérarchique en générale.

La deuxième partie est consacrée à la présentation de l'architecture interne, la conception et l'implémentation de l'agent superviseur.

ChapitreIII:Architecture interne de l'agent superviseur

Chapitre IV: Conception du modèle de procédé logiciel

Chapitre V: Implémentation et mise en œuvre

Introduction générale

CHAPITRE I

Définition et concepts de base

Introduction générale

Un modèle de procédé logiciel représente le chemin formel pour organiser et décrire de cycle de vie du logiciel, la méthodologie de développement, les outils et ressources et les développeurs qui travail dessus. **[14]**

La modélisation des procédés logiciels reste un domaine de recherche vaste ou les solutions proposées sont diverses et multiples. Malgré ces multiples propositions, ces solutions ne répondent pas toujours aux attentes des utilisateurs. Une modélisation efficace doit prendre en considération tous les paramètres de l'environnement de développement (outils, ressources, développeurs ...) les modéliser fidèlement et cela pour pouvoir anticiper les imprévus et les déviations qui peuvent se produire lors de l'exécution.

Problématique:

D'après les travaux effectués dans le génie logiciel, le concept d'agent a été utilisé particulièrement pour le développement d'environnement de programmation et précisément pour l'exécution des modèles de procédé logiciels. Les travaux se sont focalisés sur l'exploitation des capacités d'interaction et la possibilité de distribuer les tâches, par spécialisation ou par redondance des agents. Par contre, peu de travaux ont été effectués concernant la modélisation des procédés logiciels basés agents. Les capacités représentationnelles des agents sont ainsi faiblement mises a profit.

Motivation et objectifs :

Le but de ce travail est de concevoir et réaliser un agent dit « superviseur » qui a pour rôle d'assister, orienter, corriger l'utilisateur afin d'avoir un modèle de procédé qui reflète fidèlement et à tout moment l'état de l'environnement de développement du produit logiciel.

Pour réussir la conception du modèle de procédé orienté agent, nous nous somme fixés les objectifs suivants:

a- Pour que le travaille distribué soit faciliter, le modèle de procédé sera distribué, et donc fragmenter en un ensemble de fragments ou chaque fragment du modèle va s'exécuter dans un espace de travaille. La fragmentation du modèle de procédé doit être effectuée par un agent en respectant des règles qui vont être étudiés ultérieurement.

b- Représenter le modèle global ou partiel du procédé logiciel dans la base de connaissances de l'agent,

c- Respecter les spécificités locale des espaces de travail et donner une liberté de décision locale ; cela en initialisant des agent qui ont une perception de leur environnement limité a leur espace de travaille, "ce qui se passe de l'autre coté ne doit pas être leur problème".

Notre mémoire sera organisé comme suit:

- Dans le premier chapitre Nous présentons les technologies des agents et des procédés logiciels.
- Le deuxième chapitre est réservé pour la présentation de la structure et des caractéristiques de notre SMA globale, sachant que notre travail est focalisé sur le niveau supérieur de ce système.
- Le troisième chapitre concerne la présentation de l'architecture interne de notre agent superviseur.
- Le quatrième chapitre est consacré pour la conception de notre système.
- Finalement, le cinquième chapitre présente une proposition d'une implémentation notre agent superviseur.

Introduction :

Pour une meilleure compréhension des approches et technologies présentées; nous considérons qu'il est nécessaire de définir toutes les notions et concepts de bases qui vont être utilisés dans les chapitres suivants .De ce fait, ce chapitre introduit particulièrement des définitions sur : les agents et les environnements centrés procédés logiciels.

3-1-Agent et agent logiciel :

I-1-1-Définition de l'agent :

- Un agent est une entité qui perçoit son environnement et agit sur celui-ci.

- Un agent est un système informatique, situé dans un environnement, et qui agit d'une façon autonome pour atteindre les objectifs (buts) pour lesquels il a été conçu.

- Les agents intelligents sont des entités logiciels qui réalisent des opérations à la place d'un utilisateur ou d'un autre programme, avec une sorte d'indépendance ou d'autonomie, et pour faire cela ils utilisent une sorte de connaissance ou de représentation des buts ou des désires de l'utilisateur." **[9]**

- Un agent est une entité qui fonctionne continuellement et de manière autonome dans un environnement où d'autres processus se déroulent et d'autres agents existent

- Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, qui, dans un univers multi agents, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres Agents. **[4]**

- Un agent logiciel est simplement un autre genre d'abstraction de logiciel, une abstraction de la même manière que les méthodes, les fonctions et les objets sont des abstractions de logiciel. Un objet est une abstraction qui décrit des méthodes et des attributs d'un composant de logiciel .Un agent, toutefois, est une abstraction à niveau élevé de logiciel qui fournit une matière commode et puissante de décrire une entité complexe du

logiciel. Plutôt que de définir en termes de méthodes et attributs, un agent est défini en fonction de son comportement. **[18]**

- L'intelligence est définie comme un degré de raisonnement et une disposition à l'apprentissage. **[20]**

- L'agent est dit intelligent car il a un certain degré d'autonomie, des capacités décisionnelles ; en d'autres termes il a des capacités d'apprentissage.

I-1-2-Les types d'agents :

I-1-2-1-Agents cognitifs:

Agents cognitifs ont une représentation partielle de l'environnement, des buts explicites, ils sont capables de planifier leur comportement, mémoriser leurs actions passées, communiquer par envoi de messages, négocier, etc. Un SMA constitué d'agents cognitifs possède communément peu d'agents. **[12]**

I-1-2-2-Agents réactifs:

Les agents réactifs sont souvent qualifiés de ne pas être " intelligents " par eux-mêmes. Ils sont des composantes très simples qui perçoivent l'environnement et sont capable d'agir sur celui-ci. Ils n'ont pas une représentation symbolique de l'environnement ou des connaissances et ils ne possèdent pas de croyances, pas de mécanismes d'envoi de messages. Leurs capacités répondent uniquement au mode stimulus/action qui peut être considéré comme une forme de communication. Un SMA constitué d'agents réactifs possède généralement un grand nombre d'agents et présente un comportement global intelligent. **[12]**

Les agents réactifs sont considérés intelligents au niveau du groupe, du système. En conséquence, l'intelligence est distribuée entre beaucoup d'agents réactifs et le comportement intelligent devrait émerger de l'interaction entre ces agents réactifs et l'environnement.

I-1-3-comparaison entre un agent réactif et un agent cognitif: [7]

Caractéristique d'un système d'agents cognitifs	Caractéristique d'un système d'agents Réactifs
Présentation explicite de l'environnement	Pas de représentation
Peut tenir compte son passé	Pas de mémoire de son histoire
Agents complexes	Fonctionnement stimulus/réponse
Petit nombre d'agent	Grand nombre d'agents

Tableau I-1 : comparaison entre un agent réactif et un agent cognitif

I-1-4-Propriétés d'un agent :

Pour qualifier qu'une entité (physique ou abstraite) est un agent /agent intelligent ou non, il faut que cet agent possède certaines propriétés, qui sont **[2]**:

a- L'autonomie:

L'agent peut agir sans l'intervention directe de l'être humain (ou autres agents) ; ainsi il peut planifier des étapes d'exécution de taches ou assigner des ressources de manière automatique sans une intervention externe. **[1]**

b- Des capacités sociales :

Cela veut dire que l'agent a les moyens pour interagir, communiquer et échanger des informatiques avec des personnes ou d'autres agents. **[8]**

c- Des capacités d'apprentissage et d'adaptation :

L'agent peut s'adapter avec les changements dynamiques de son environnement ; par exemple : il peut mettre a jour sa base de données ou bien ajouter d'autres Règles pour pouvoir agir selon l'état actuel du système. **[8]**

d- Des capacités décisionnelles :

Décide des actions à entreprendre selon le but à réaliser et selon l'état de l'environnement. **[13]**

Ces trois caractéristiques ne sont pas forcément présentées en même temps dans un agent, les agents qui possèdent les trois caractéristiques sont dits agents intelligents idéaux comme la représente la figure suivante:(figure1-1).

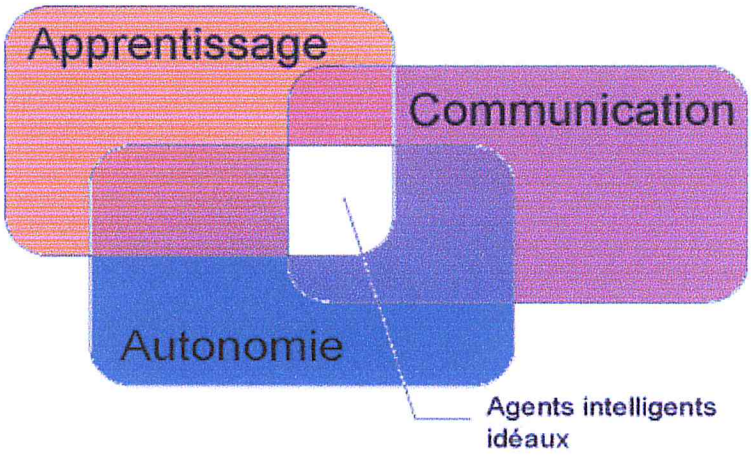


Figure I-1 : caractéristiques d'un agent intelligent

composants

I-1-5-Architecture d'un agent cognitive:

Nous présentons l'architecture générale d'un agent cognitive (figure1-2) **[18]**

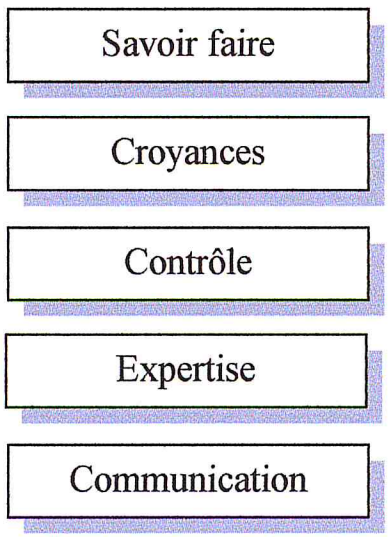


Figure I-2: Architecture d'un agent cognitive

a- Savoir faire :

Le savoir faire est une interface qui permet la sélection des agents a solliciter pour atteindre ses objectif, et aussi permettant la déclaration des connaissances et des compétences de l'agent **[18]**

b- Croyances :

Les Croyances d'un agent constituent les connaissances que l'agent possède sur lui même, sur des autre agents et même sur les environnement .Ces connaissances ne sont pas nécessairement objectives ou vrais, peuvent être incorrectes ou incomplètes .les logiques de connaissances et de croyances s'intéressent à la formalisation de quelques connaissances considérées comme incertaines .cette formalisation détermine une grande partie de comportement intelligent d'un agent. **[18]**

c- Contrôle :

Afin d'atteindre ses objectifs ,l'agent contrôle tout changement dans son comportement et leur environnement .les buts ,les intentions ,les plans et les tâches qu'il possède ,représentent leur connaissance de contrôle. **[18]**

d- Expertise :

C'est une connaissance qui permet à l'agent de résoudre un problème .par exemple dans les systèmes experts les règles de production présentent ce type de connaissances. **[3]**

e- Communication :

L'interaction entres agents est possible grâce a un protocole de communication pour assurer une bonne coopération et coordination d'action. En plus, l'agent peut utiliser d'autre connaissance de communication liée au réseau de communication. **[3]**

I-1-6-Difinition d'un SMA:

- Un système multi agent est un ensemble organisé d'agents.
- Un SMA est constituée de résolves (agents) qui travaillent ensemble pour résoudre un problème donné. **[4]**

Les avantages les plus importants sont:

a- La décentralisation :

Il est capable de découper un système complexe en un ensemble de sous systèmes décentralises coopérants, de plus, plusieurs groupes de l'organisation peuvent être distribuées géographiquement. [9]

b- Réutilisation des composants ou sous systèmes précédents:

les SMAs nous donnent cet avantages:le recyclage des composantes , ce qui nous permet un gain de temps et la création de nouveaux systèmes en inter-opérant ceux déjà existants même si le degré d'hétérogénéité est assez élevé. [9]

c- Support de travail coopératif:

Il est capable de modéliser et supporter une large gamme d'interaction dans le travail coopératif, pour ce la les agents logiciels peuvent interagir en étant autonomes de toute assistance humaine. [9]

d- Flexibilité: (souplesse)

Il est capable de prendre en charge les caractéristiques d'un environnement distribué des systèmes hétérogènes, dans une évolution permanente [21].

I-1-7- les caractéristiques d'un SMA:

- Chaque agent a des informations ou des capacités de résolution de problèmes limités (ainsi .chaque agent a un point de vue partiel);
- Il n' y a aucun contrôle globale du système multi -agent.
- Les données sont décentralisées.
- Le calcule est asynchrone. [6]

I-1-8- Communications inter agents:

Les communications, dans les SMA comme chez les humains, sont à la base des interactions et de l'organisation. Une communication peut être définie comme une forme d'action locale d'un agent vers d'autres agents. Les questions abordées par un modèle de communication peuvent être résumées par l'interrogation suivante : qui communique quoi, à qui, quand, pourquoi, et comment, pourquoi les agents communiquent-ils ? La communication doit permettre la mise en œuvre de l'interaction et par conséquent la coopération et la coordination d'actions.

Quand les agents communiquent-ils ? Les agents sont souvent confrontés à des situations où ils ont besoin d'interagir avec d'autres agents pour atteindre leurs buts locaux ou globaux. La difficulté réside dans l'identification de ces situations. Par exemple, une communication peut être sollicitée suite à une demande explicite par un autre agent.

Avec qui les agents communiquent-ils ? Les communications peuvent être sélectives sur un nombre restreint d'agents ou diffusées à l'ensemble des agents. Le choix de l'interlocuteur dépend essentiellement des accointances de l'agent (connaissances qui a l'agent sur les autres agents).

Comment les agents communiquent-ils ? La mise en œuvre de la communication nécessite un langage de communication compréhensible et commun à tous les agents. Il faut identifier les différents types de communication et définir les moyens permettant non seulement l'envoi et la réception de données mais aussi le transfert de connaissances avec une sémantique appropriée à chaque type de message. **[12]**

I-1-9-Interactions entre agents:

Une des principales propriétés de l'agent dans un SMA est celle d'interagir avec les autres agents. Ces interactions sont généralement définies comme toute forme d'action exécutée au sein du système d'agents et qui a pour effet de modifier le comportement d'un autre agent. Elles permettent aux agents de participer à la satisfaction d'un but global.

Cette participation permet au système d'évoluer vers un de ses objectifs et d'avoir un comportement intelligent indépendamment du degré de complexité des agents qui le composent.

En général, les interactions sont mises en œuvre par un transfert d'informations entre agents ou entre l'environnement et les agents, soit par perception, soit par communication. Par la perception, les agents ont connaissance d'un changement de comportement d'un tiers au travers du milieu. Par la communication, un agent fait un acte délibéré de transfert d'informations vers un ou plusieurs autres agents. L'interaction peut être décomposée en trois phases non nécessairement séquentielles

- La réception d'informations ou la perception d'un changement,

- Le raisonnement sur les autres agents à partir des informations acquises,

- Une émission de message(s) ou plusieurs actions (plan d'actions) modifiant l'environnement. Cette phase est le résultat d'un raisonnement de l'agent sur son propre savoir-faire et celui des autres agents.

Le degré de complexité des connaissances nécessaires pour traiter les interactions dépend des capacités cognitives (de raisonnement) de l'agent et du fait que l'agent a des connaissances ou non de l'objectif du système global. En effet, un agent qui poursuit un objectif individuel au sein du système comme c'est le cas pour les agents dits réactifs ne focalise pas son énergie pour interagir avec les autres même s'il y est amené. Par contre, un agent qui participe à la satisfaction du but global du système tout en poursuivant un objectif individuel, va passer une partie de son temps à coopérer ou à se coordonner avec les autres agents. Pour cela, il doit posséder des connaissances sociales qui modélisent ses croyances sur les autres agents. **[12]**

I-2- Les procédés logiciels et les environnements centrés procédés :

I-2-1- Définition du modèle de procédés logiciels:

Un procédé logiciel est un ensemble d'activités ,de règles , procédures, technique, outils et ressources faisons intervenir un ensemble de personnes (développeurs) afin d'assurer le développement la maintenance d'un produit logiciel et cela bien entendu dans les meilleurs délais avec le meilleur coût pour un produit de meilleure qualité possible. **[21] [14]**

Un modèle de procédé logiciel représente le chemin formel pour organiser et décrire le cycle de vie du logiciel, la méthodologie de développement, les outils et ressources et les développeurs qui travail dessus. **[14]**

I-2-2-Définition de l'environnement centré procédé logiciels (ECP):

Un ECP est un système ou les procédés logiciels sont modélisés et exécutés. **[4]**

Un ECP est un système dans lequel le logiciel est fabriqué selon son modèle de procédé **[14]**

I-2-3-Les composants d'un ECP:

a- Pilote d'exécution de procédé *:(process enactment driver) : ou bien moteur d'exécution:*

Le PEP permet l'exécution ou l'interprétation d'un MPL (modèle de procédé logiciel) selon ses activités hiérarchique ; il gèrent l'ordre des sous tâches qui doivent être effectuées et les conditions à satisfaire avant leurs application ; il joue le rôle d'un pilote automatique qui initialise le MPL, exécute ses sous tâches, accepte les données d'entrées d'un développeur, met a jour et diffuse l'état de ses sous tâches et leurs déclenchement.

L'état d'un MPL ou d'une tache est l'indicateur de l'état actuel de développement ; l'état est Mis à jour par le pilote d'exécution de procédé et il est basé sur l'interaction avec les développeurs. La mise à jour de l'état avec l'état actuel de développement représente la progression dans l'exécution du projet [16].

b- Interface développeurs :

L'interface développeur est un environnent qui montre le MPL et ses activités ; cette interface permet aux différents développeurs d'exécuter les MPLs de manière concurrente et effectuer seulement les sous taches dont ils ont besoin à travers d'un processus d'orientation (ou de guidage) et des espaces de travail. Le processus de guidage est un moyen d'informer les développeurs quels sont les sous taches a effectuer et quand ces sous taches sont prête a commencer.

L'interface développeurs est constituée de plusieurs < fenêtre de tache > qui représentent l'état de progression des taches et leurs sous taches sous forme explicite < graphes, formes géométriques>. **[16]**

c- Base d'objet :

Cette base contient toutes les informations qui concernent le développement du logiciel, les versions développées, les documents le concernant, les modèles de procédés utilisés...etc.

Le gestionnaire des bases d'objets prend en charge tous ce qui concerne la base d'objets : le stockage des information, les droits d'accès, l'accès concurrent ...etc. **[4]**

d- Espaces de travail :

C'est un lieu (peut être réel ou virtuel) ou travail un groupe de développeurs, tous les moyens, outils de travail et de communications sont fournis pour avoir un bon environnement de développement . **[4]**

e- Serveur de communication :

C'est un élément important dans l'environnement centré procédé car c'est lui qui gère toute transaction, échange d'informations entre les espaces de travail, les développeurs que ce soit communication, négociation ou autre. **[4]**

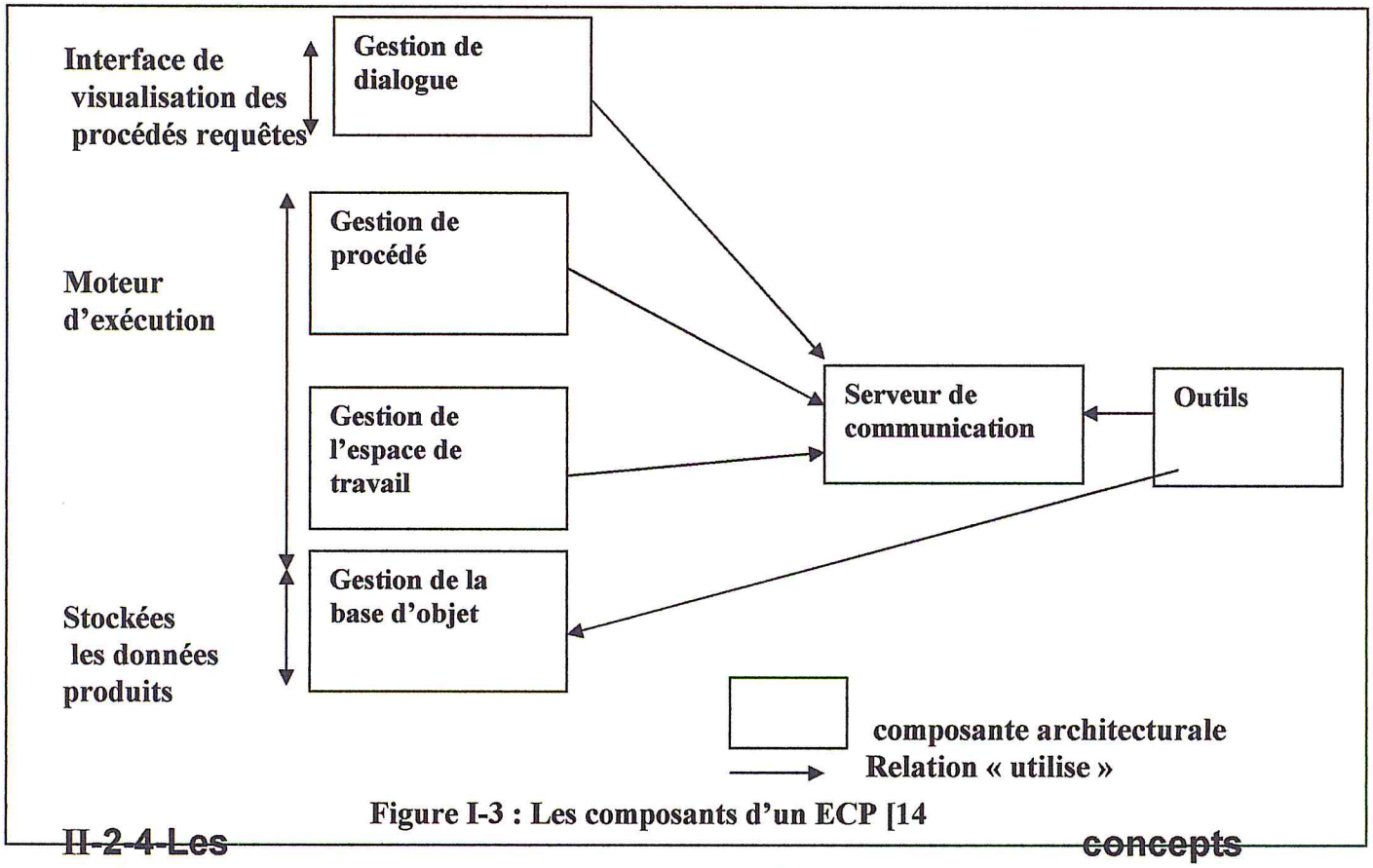


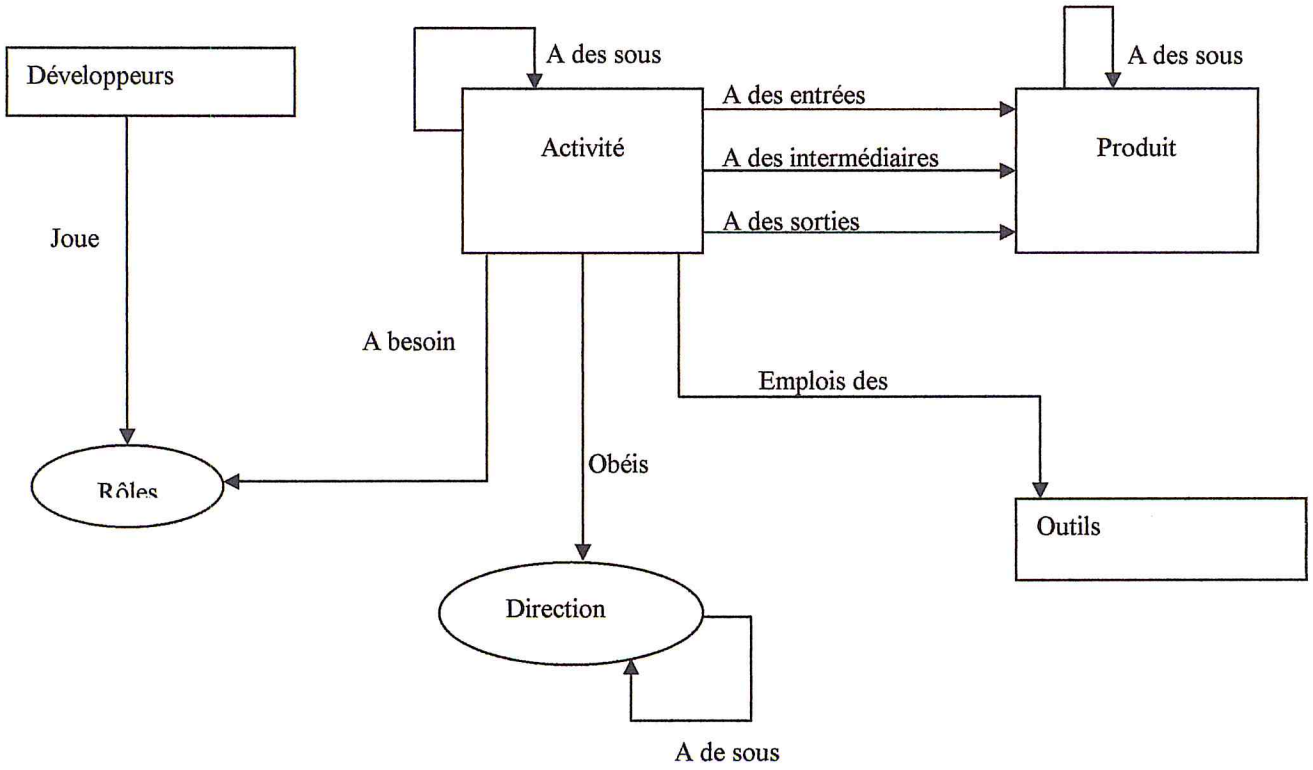
Figure I-3 : Les composants d'un ECP [14

II-2-4-Les concepts de base pour la modélisation des procédés logiciels:

Pour modéliser les procédés logiciel on est tenté de connaître les concepts de bases de la modélisation du procédé logiciel qui sont:

- **Activité :** Est une opération atomique ou une étape de procédé.
- **produits :** L'ensemble des objets, programmes à développer, délivrer, maintenir dans un projet.

- **ressources :** Ce sont les moyens exigés par l'activité pour être exécutée, il y a deux types de ressources : Humains, matériels.
- **Le développeur :** (agent humain) généralement directement lié à l'activité par son rôle.
- **Le rôle :** peut être utilise pour définir les responsabilités et obligations du développeur.
- **La direction :** c'est le pilote <<guide >> qui peut être des Règles ou des procédures qui gouverne l'activité.
- **Les outils :** C'est l'équipement matériel ou logiciel dont l'activité a besoin pour son exécution.



FigureI-4 : Le concept de base pour la modélisation des procédés logiciels

II-2-5-langage de modélisation de procédés [14] [20]:

PML basé langage de programmation :

Il décrit le procédé sous forme de programme. Pour cela, il utilise des langages de programmation classiques étendus par des concepts relatifs aux procédés.

PML basé règles :

Dans cette approche, le procédé est décrit par règles pré- condition / action/ post-condition. La pré- condition est une contrainte nécessaire à l'exécution de l'action tandis que la post-condition représente l'effet de la condition.

PML basé réseau :

Le procédé est décrit par un réseau de pétri ou les transitions représentent les activités et les places les prés- conditions et post-conditions. La vérification d'une condition est représentée par la présence d'un jeton dans cette place.

PML basé multi- paradigmes :

Un seul formalisme peut difficilement décrire tous les éléments du procédé logiciel ainsi que toutes les étapes. En effet, un formalise qui décrit < tout > ne peut être que très complexe et surchargé par les concepts utilisé. L'approche multi- paradigme combine deux ou plusieurs paradigmes pour décrire les différents éléments et activités du procédé logiciel.

II-2-6-Les aspects d'un ECP efficace :

- Un ECP peut être géographiquement distribué, tel que chaque groupe de travail peut être dans un lieu distant (une autre ville ou un autre pays), il peut développer un composant a n'importe quel moment de la journée sans se soucier des autres groupes de travail.

- Les modèles de procédé peuvent être hétérogènes, un environnement d'ingénierie logiciel doit être capable de supporter tout type de procédé logiciel.

- L'interaction, communication, négociation, collaboration prennent une place très importante et doivent être mis en place d'une manière efficace.

- Un ECP doit pouvoir utilisé tout type d'outils.

- Un ECP doit gérer l'évolution dans des modèles de procédé logiciel et doit supporter leur changement dynamique pendant l'exécution. **[2]**

- Un ECP doit garder la cohérence des informations manipulées.

Conclusion :

D'après la plupart des travaux effectués dans le domaine des SMA, les caractéristique de base de l'agent sont: L'autonomie, la sociabilité et la réactivité, et d'après notre étude sur les procédés logiciels il est évident que l'agent peut répondre aux besoins et aux exigences des procédés logiciels.

CHAPITRE II
Présentation du SMA

Introduction

Dans le but de vous donner une vue globale sur notre travaille nous nous proposons dans ce qui suit une petite présentation d'un modèle de procédé sous forme d'un SMA hiérarchique doté d'outils, modèles et connaissances nécessaires.

II-2- Les agents du modèle de procédé logiciel

Notre SMA se compose de trois catégories d'agents le premier est le niveau supérieur dit "AGENT SUPERVISEUR", le deuxième est le niveau intermédiaire dit "AGENT FRAGMENT", et le dernier c'est le niveau inférieur dit "AGENT TACHE" tel que notre travail ne s'intéresse qu'au niveau supérieur du SMA .

II-2-1- Agent superviseur:

Son rôle est de modéliser et de stocker le procédé logiciel global, le fragmenter et l'affecter aux agent fragment dans les différents espace de travail pour qu'il s'exécutent, il a une vue globale de l'exécution des activités de modèle de procédé dans tous les espaces de travail ; ainsi son rôle est de garder la cohérence et la consistance du procédé logiciel.

L'agent superviseur est en interaction avec le chef de projet<humain>; ainsi il est possible à l'être humain d'avoir une vue globale de l'exécution du modèle de procédé ou d'introduire des changements même pendant l'exécution.

II-2-2-Agent fragment:

Cet agent est localisé dans un espace de travaille et possède le fragment de procédé logiciel qui doit s'y exécuter.

Son objectif est de gérer l'exécution de son fragment de procédé ; son rôle consiste à assigner les activités du modèle de procédé sur les différents agent tâche qu'il initialise pour l'exécution de ces tâches.

II-2-3-Agent tâche

Son rôle est de réaliser les activités du modèle de procédé qu'on lui a assigné ; de fournir les ressources requises, soit directement si elles sont disponibles à son niveau ou par demande de service ou négociation dans le cas contraire, et de les exécuter. A la fin de l'exécution un rapport d'exécution est rendu à l'agent fragment.

Voici un diagramme qui représente notre SMA d'une manière générale.

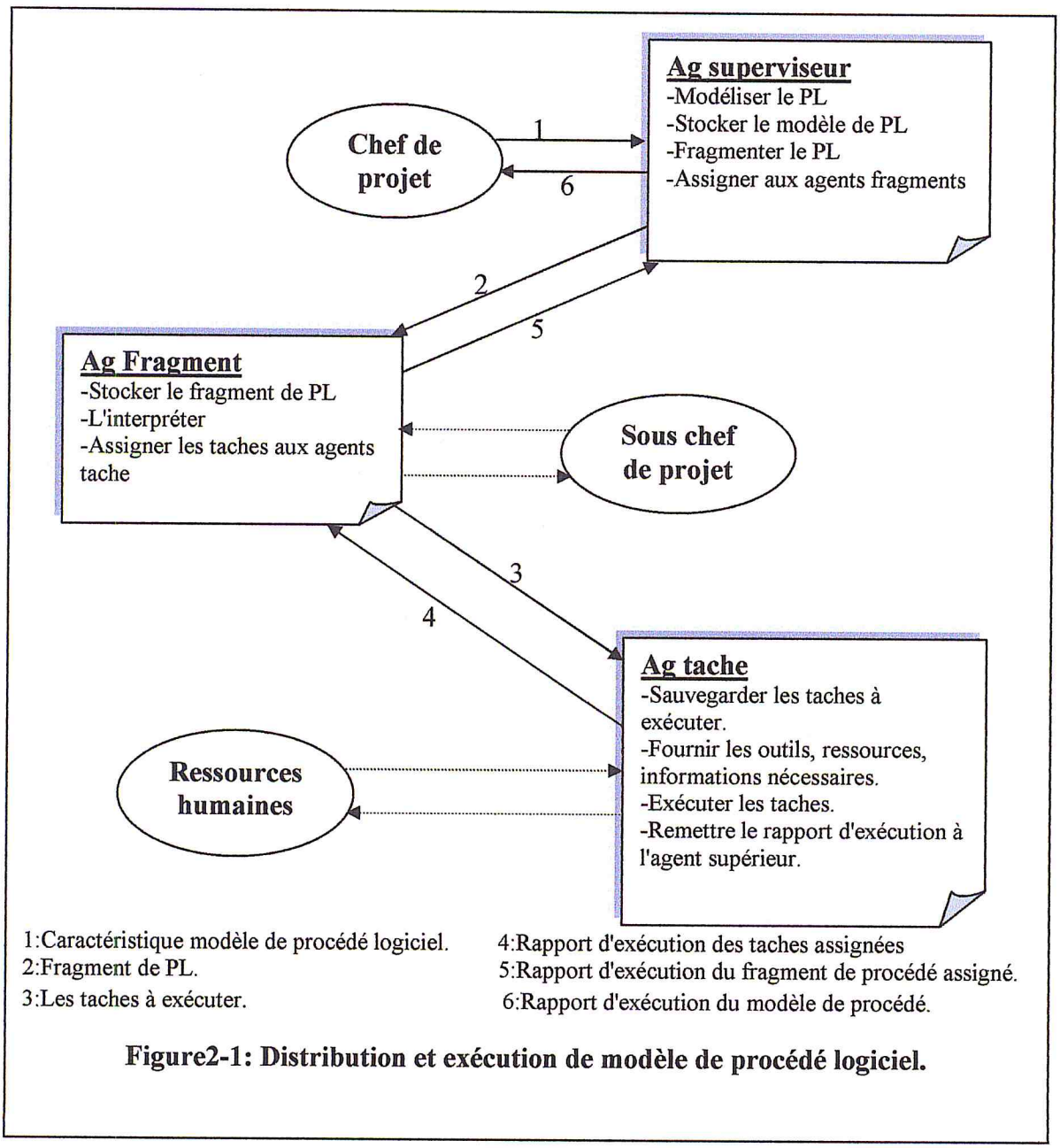


Figure2-1: Distribution et exécution de modèle de procédé logiciel.

II-3-Les principales fonctions de l'agent superviseur

II-3-1-Modélisation des procédés logiciels:

L'agent superviseur fournit une interface qui permet la modélisation de procédé logiciel en interaction avec le chef de projet, l'interface permet au chef de projet de faire rentrer Le modèle de procédé logiciel activités par activité en utilisant des ressources des technique et des facteurs d'évaluation pour réaliser et atteindre notre objectif et de fournir des condition nécessaire a son exécution.

II-3-2-Etablir les facteures d'evaluation:

Pour verifier que le procédé modélisé est biensuivé lors de l'exécution ainsi il est possible de prendre des décision de changements s'il y à un retard lors de l'exécution on a choisit d'établir des facteurs d'évaluations, ces facteurs sont: le temps et le coût tel que ces facteurs sont estimés :

Pour estimer ces deux facteurs on a adopté le modèle d'estimation COCOMO décrit par Boehm, il existe sur deux versions :

- Le modèle COCOMO simple suppose que les besoins ne seront pas significativement modifiés pendant le développement du système et aussi que le déroulement du projet est suivi à la fois par le client et par l'organisation qui le développe, dans cette version de COCOMO les formules permettent de calculer le coût, ou plus exactement l'effort hommes machines sont **[22] :**

-mode organique : $HM = 2.4 (KLSL)^{1.05}$.

-mode Semi détaché : $HM = 3(KLSL)^{1.12}$.

-mode mode embarqué : $HM = 3.6 (KLSL)^{1.20}$.

Pour calculer le temps de développement en utilise les formules suivants :

-mode organique : $TDEV = 2.5 (HM)^{0.38}$

- mode Semi détaché : $TDEV = 2.5 (HM)^{0.35}$

-mode embarqué : $TDEV = 2.5 (HM)^{0.32}$

HM : est le nombre d'hommes- mois nécessaire à la réalisation du projet.
KLSL : est le nombre de Kilo lignes- Sources Livrées.

- Pour la deuxième version de ce modèle en trouve le COCOMO intermédiaire qui utilise des attribues du personnel et des attribues du projet :

1-Les attribues de projet sont :
-FIAB : fiabilité requise du logiciel
-DONN : Taille de la base de données
-COLX : complété de produit

2-Les attributs de l'environnement matériels et logiciel sont :
-TEMP : Contraintes de temps d'exécution.
-ESPA : Contraintes d'espaces mémoire.
-VIRT : Volatilité de la machin virtuelle :
-CSYS : Contraintes du système de développement.

3-Les attributs du projet sont :
-PMOD : Méthodes de programmations modernes.
-OLOG : Outil logiciel.
-DREQ : Duré requise du développement

On peut résumé les attribues précédent dans la table suivant qui indiquera clairement et de manière complète les multiplicateurs associes a ces attributs, notons que :
TB : signifie très bas.
B : signifie bas.
M : moyen.
E : élevé.
TE : très élevé.

Attributs	TB	B	M	E	TE
FIAB	0.75	0.88	1.00	1.15	1.40
DONN	--	0.94	1.00	1.08	1.16
CPLX	0.70	0.85	1.00	1.15	1.30
TEMP	--	--	1.00	1.11	1.30
ESPA	--	--	1.00	1.06	1.21
VIRT	--	0.87	1.00	1.15	1.30
CSYS	--	0.87	1.00	1.07	1.15
APTA	1.46	1.19	1.00	0.86	0.71
EXPA	1.29	1.13	1.00	0.91	0.82
APTP	1.42	1.17	1.00	0.86	0.70
EXPV	1.21	1.10	1.00	0.90	--
EXPL	1.14	1.07	1.00	0.95	--
PMOD	1.24	1.10	1.00	0.91	0.82
OLOG	1.24	1.10	1.00	0.91	0.83
DREQ	1.23	1.08	1.00	1.04	1.10

TableII-1: les multiplicateurs associes aux attributs

II-3-3-Stocker le modèle de procédés dans la base de connaissance

Lorsque l'agent superviseur termine la modélisation de procède logiciels ; il doit stocker ce dernier dans sa base de connaissance, cette base doit contenir toutes les informations concernant le procédés logiciels.

II-3-4-La fragmentation :

Une des solutions les plus sensées et de fragmenter le procèdes en respectant le cycle de vie de logiciel comme il est illustré dans la figure suivante:

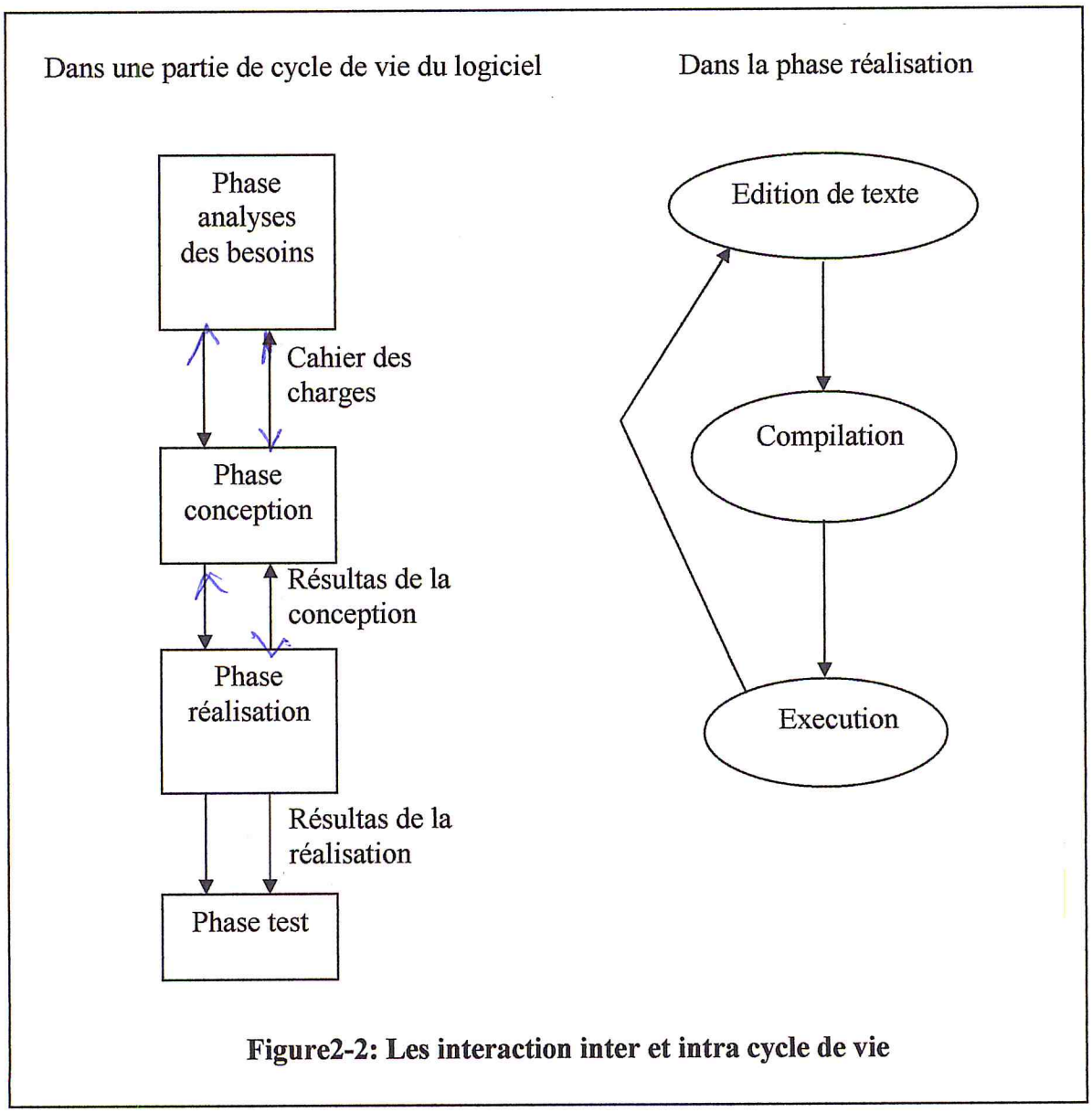


Figure2-2: Les interaction inter et intra cycle de vie

Les interactions entre les phases du cycle de vie se limitent au passage des résultats d'une phase à la phase suivante ; par contre les interactions internes d'une phase sont plus denses ; par exemple, dans la figure 2-2 les interactions entre les taches <<l'édition des textes >>, << Compilation >> et <<exécution >>sont plus fréquentes que celle entre les phases.

L'autre technique de fragmentation est de respecter les spécialités des espaces de travail et de fragmenter le modèle de procédé logiciel selon les ressources et outils existants dans des espaces de travail illustré dans la figure2-3:

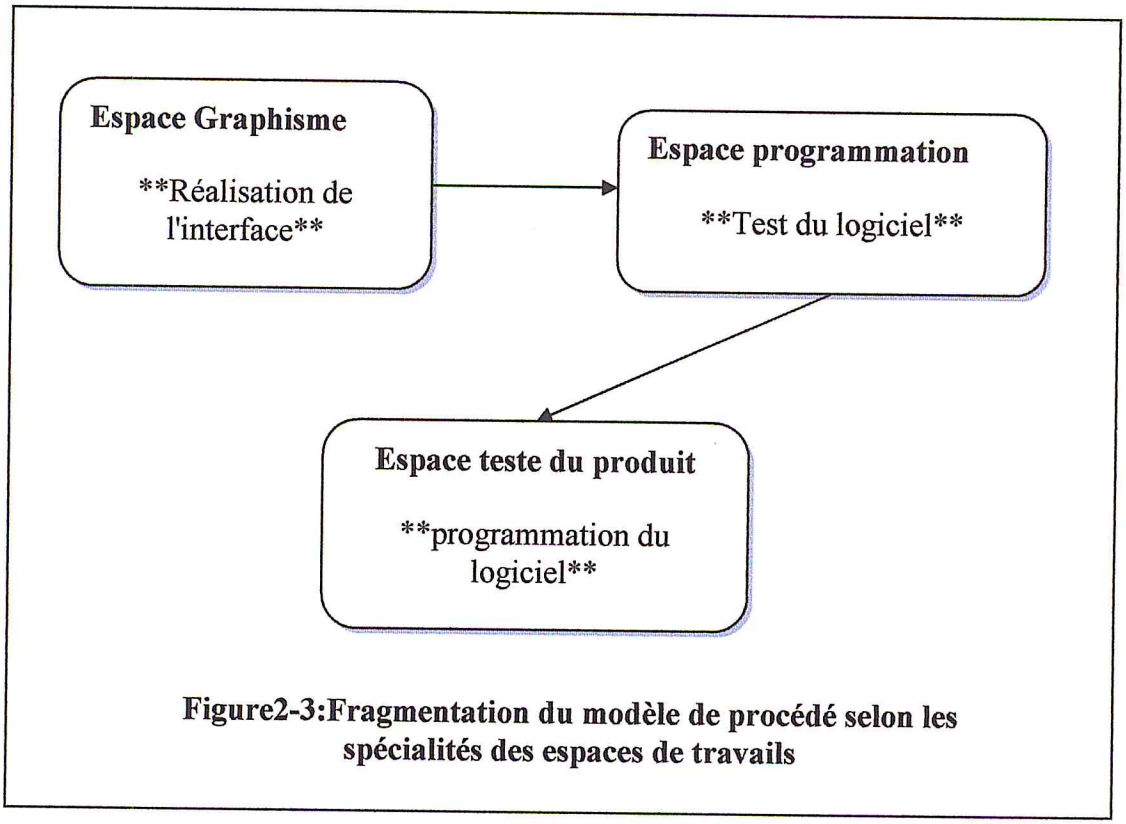


Figure2-3:Fragmentation du modèle de procédé selon les spécialités des espaces de travails

II-3-5-Initialisation des agent fragment:

Cette fonction consiste à créer un agent fragment et lui affecter un des fragments obtenus comme résultat de la fonction précédente "fragmentation des PL".

Conclusion:

Dans ce chapitre nous avons présenté les principales fonctions de notre agent superviseur ainsi une présentation générale du systèmes multiagents. notre travail d'implémentation se base sur l'agent supervisur qui est l'objet de notre projet et pour cela on passe dans le chapitre suivant à représenter l'architecture interne de notre agent superviseur.

Introduction :

L'architecture modulaire est conçue comme un assemblage de modules ou chacun de ceux-ci réalise une fonction horizontale particulière .cette architecture convient parfaitement à notre agent modèle de procédé ou sont exécutées plusieurs fonction horizontales en parallèle ; elle nous permet aussi de prédéfinire les échanges des informations entre ces modules et nous donne ainsi un moyen de définir le comportement de l'agent face aux événements qui peuvent se produire.

L'architecture a base de règles ne convient pas à notre agent modèle de procédé, en effet, La nature de notre modèle de procédé logiciel ne nous permet pas d'avoir en permanence une base de règles pertinente dans la mise ajour régulière et indispensable peut nuire a la qualité de fonctionnement de notre agent ; par ailleurs ; une architecture interne identique a tout les agents du modèle de procédé présent le grand avantage de ne pas avoir à programmer individuellement tous les types d'agent.

III-1- Architecture interne de l'agent superviseur :

L'architecture de agent est modulaire horizontale ; chaque agent est composé de 6 modules qui travaillent en parallèle et en collaboration .chaque module est responsable de certaines taches internes de l'agent, la représentation de l'architecture de l'agent modèle de procédé logiciel peut être comme suite :

L'agent modèle de procédé est composé de deux types de modules :

- Les modules gestionnaires de l'état interne de l'agent : le rôle de ces module est la gestion du fonctionnement interne de l'agent .Ce sont les modules <<communication>> ; << Module de gestionnaire de la base de connaissances>> ; et <<gestionnaire d'accointances>>.

- Les autres modules sont des modules de modélisation et d'exécution du modèle de procédé ; leur rôle principal est la gestion de procédé logiciel aux niveaux de la modélisation, de la planification, de l'exécution. Ces modules sont :<< Module Gestionnaire De procédés logiciels>> ;<< Module de décision et de planification>> ;<< Module de Réalisation et d'évaluation >>.

III-1-1-Module de communication :

Ce module s'occupe essentiellement de la communication avec l'environnement externe.

L'environnement externe peut être composé d'agents logiciels ou d'agents humains.

Les fonctions essentielles de ce module sont l'émission et la réception des messages.

Lors de la réception d'un message, le module définit le type de message, identifie le module destinataire concerné et lui transmet le message.

Le parallélisme des modules se matérialise et apparaît lors des réceptions des messages.

Le module de communication peut recevoir plusieurs messages simultanément destinés aux différents modules de l'agent, ces derniers peuvent recevoir leurs messages internes et travailler en parallèle sans difficulté.

On peut sites quelques messages reçus et envoyer par le module de communication :

message	contenu
Consignes de modélisation	Tache, objectifs, ressources, technique, facteurs d'évaluations, ...etc.
Fragment assigné	Le fragment du modèle de procédé logiciel assigné par l'agent a un agent fragment
Messages d'interactions	Ils contiennent des messages de négociation ou de demande de services, tous ce qui concerne une coopération entre agents.
Rapport d'exécution	Rapport qui contient les résultats de l'exécution.
Changements de valeurs d'évaluations	Ce message est toujours envoyé par l'agent superviseur à l'agent fragment. Les changements sont toujours pour élargir les valeurs d'évaluations.

Tableau III-1 : les messages reçus et envoyer par le module de communication

III-1-2-Module gestionnaire de la base des connaissances :

Ce module gère la base de connaissance de l'agent superviseur, en terme d'accès et cohérence des informations stockes sachant que cette base est accessible par la plupart des modules, cette base contient les informations suivantes :

- Le modèle de procédé logiciel et tout la version historique utilisée par cet agent.
- Le modèle d'interaction.
- Les stratégies de négociations et de collaboration suivie par cet agent ainsi que leur historique d'utilisation.
- Les paramètres d'évaluation d'exécution actuelle et les versions précédentes.
- Les rapports et résultats d'exécution et assignations.
- Les rapports des changements.
- L'état actuel des ressources.

II-1-3-Module gestionnaire d'accointances et des ressources :

Ce module veille à la mise a jour des états internes de l'agent et de ses voisins.

L'état interne de l'agent illustre l'état de ses ressources (alloué, libre, en panne ….), de ses objectif actuels ainsi ses contrats avec ses voisins.

Le module gestionnaire d'accointances et des ressources est le premier informé par le module communication d'une libération de ressources.

II-1-4- Module de décision et de planification :

Ce module est considéré comme le cerveau de l'agent, c'est cette partie qui s'occupe de prendre des décisions concernant les négociations en cours, les tâches à exécuter et de leur ordre d'exécution.

Tous les autres modules sont à son service, ils doivent lui fournir toutes les informations nécessaires pour prendre les bonnes décisions.

Ces décisions portent entre autres sur les allocations de ressource et l'ordre d'exécution des tâches.

II-1-5- Module réalisation et évaluation de la réalisation :

L'objectif de ce module est d'exécuter les tâches planifiées par le module précédent ; ces tâches peuvent, par exemple, être les activités du modèle de procédé logiciel.

L'avantage principal de ce module est la possibilité d'introduire le modèle de procédé partie par partie.

III-1-6- Module gestionnaire du procédé logiciel :

Le rôle principal de ce module est de formaliser le modèle de procédé de la meilleure manière possible à partir des informations reçue par le chef de projets. il doit permettre d'introduire les changements décidés par le module <<réalisation et évaluation de la réalisation>>et vérifier la consistances de ces modifications par rapport au modèle de procédé global de l'agent.

Ce module prend également en charge le suivi de l'exécution du modèle de procédé en fixent la partie à exécuter avant de faire passer à la l'agent<<décision et planification>> ainsi, si une partie sélectionnée, elle doit être exécuter même si elle est concernée par un rapport de changement.

La figure suivante (Figure4-1) montres les activités internes de l'agent superviseur, ainsi nous pouvons constater que les fonctions principales du module <<réalisations et évaluation de la réalisation>> sont l'initialisation des agents fragments, assignations des fragments de procédé logiciel et l'évaluation des résultats reçus. le module <<gestionnaire du modèle de procédé logiciels >>est le module le plus surchargé ;il à devoir de modéliser tout procédé logiciels.

III-2- Exemple d'un pseudo Algorithme de fonctionnement de l'agent superviseur :

Dans cette partie nous présentons Une description générale sur le fonctionnement de l'agent superviseur, citons que les modules de l'agent s'exécutent en parallèle.

❖ Module de communication

Début :

Si message_ reçu=<<consigne de modélisation de procédé logiciels>>
Alors 1-envoi message_ interne =<<consigne de modélisation de procédé logiciels>>
Au Module gestionnaire modèle de procédé logiciel.

2-Aller à :**Etiq0** ;
Fin de Si.

Si message_ reçu=<<message d'interaction>>
Alors :
Si message d'interaction=<<Ajout de ressources>>
Alors : 1-envoi message_ interne =<< Ajout de ressources>>
Au Module gestionnaire des accointances et de ressources.

2-Aller à : **Etiq11;**
Fin de Si.

Sinon ;
Si message d'interaction=<< libération de ressources>>
Alors : 1-envoi message_ interne =<< libération de ressources>>
Au Module gestionnaire des accointances et des ressources.

2-Aller à : **Etiq11;**

Sinon Si message d'interaction=<<Demande ou offre de services>>
Alors : 1-envoi message_ interne =<< demande, offre, négociation de service>>
Au Module de décision et planification des tâches

2-Aller à : **Etiq14;**
Fin de Si ;
Fin de Si ;
Fin de Si ;

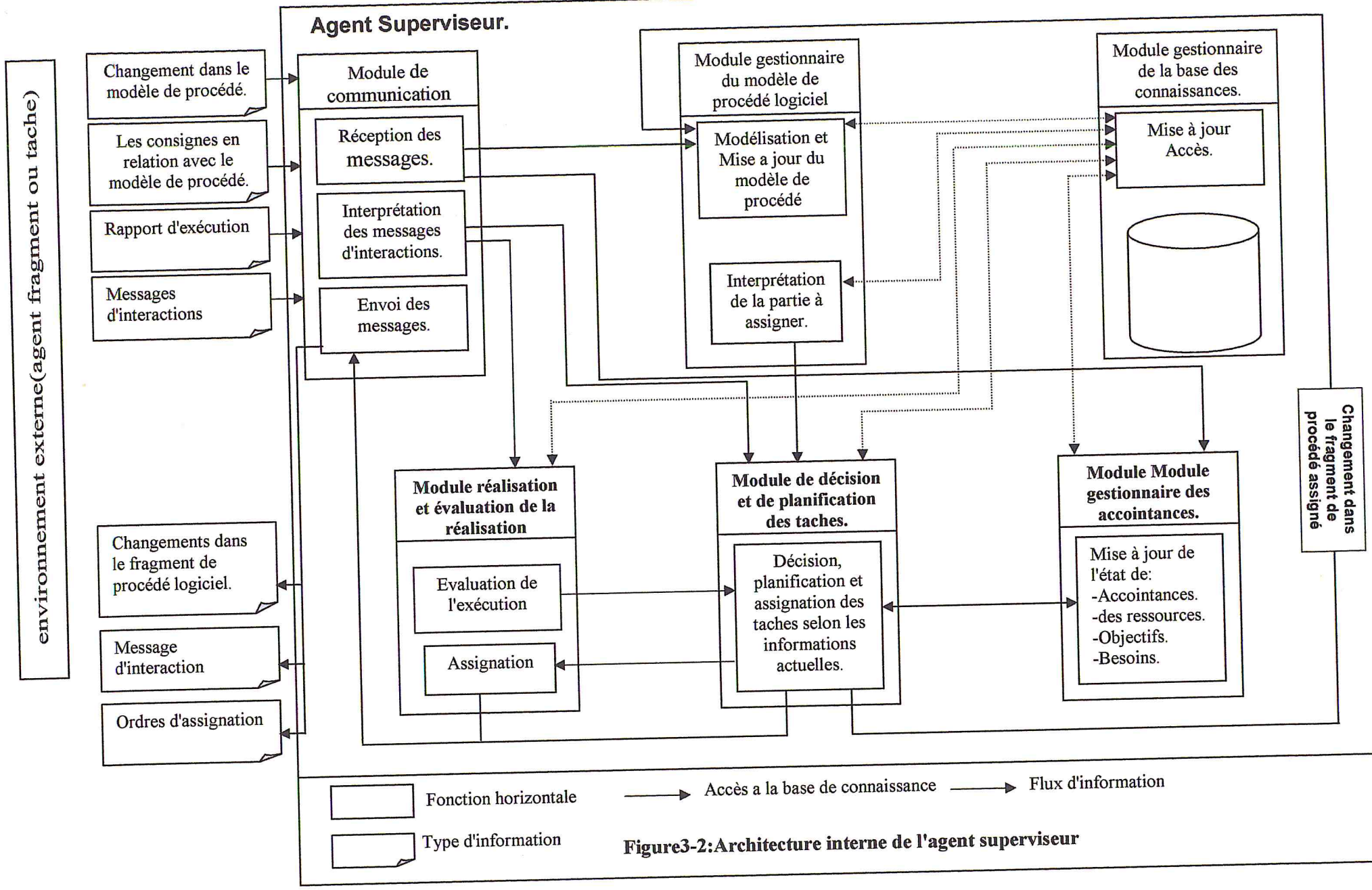


Figure3-2:Architecture interne de l'agent superviseur

Si message_ reçu=<<demande de consultation de résultats d'exécution>>
Alors :
1-envoi message_ interne =<< demande de consultation de résultats d'exécution>>
Au Module gestionnaire de la base des connaissances.

2-Aller à : **Etiq24;**
Fin de Si ;
Etiq20 : 1-envoi message_ interne<<rapport d'exécution>>
Au module réalisation et évaluation de la réalisation
2-Aller à : **Etiq21;**

Etiq6 : envoi message=<<Assignation de fragments et facteurs d'évaluations>>.
Destination=<<tout les agents fragment>>.

Etiq9 : envoi message=<<changement effectué >>.
Destination=<< agent fragment qui a effectué les changements>>.

Etiq13 : envoi message=<<MAJ d'état de ressources effectué>>.
Destination=<<chef de projet>>.

Etiq15 : envoi message=<<Réponse message d'interaction>>.
Destination=<<source de messages d'interaction>>.

Etiq22: envoi message=<<proposition des changements dans le modèle de procédé logiciel>>.
Destination=<<chef de projet>>.

Etiq25 : envoi message=<<résultat finale ou partiel de l'exécution du modèle de procédé logiciel>>.
Destination=<<chef de projet>>.

Fin.

❖ Module gestionnaire du modèle de procédé logiciel :

Début :
Etiq0: 1-modéliser le modèle de procédé logiciel en interaction avec le chef de projet jusqu'à Modélisation finale

Fin ;

❖ Module de gestion de la base de connaissance :

Début :

Etiq1: 1- sauvegarde du fragment modèle de procédé global et des facteurs d'évaluation de l'agent superviseur.

2- envoi message_ interne =<<fragment modèle de procédé prêt à être exécuter>>
Au Module gestionnaire du modèle de procédé logiciel.

3-Allez à : etiq3 ;

Etiq8: 1-sauvegarde des changements du modèle de procédé globale et des facteurs d'évaluations de l'agent superviseur.

2-envoi message_ interne =<< modèle de procédé prêt à être exécuter>>
Au Module gestionnaire du modèle de procédé logiciel.

3-Allez à : etiq3

Etiq12: 1- sauvegarde des mise à jour des états des ressources.

Si MAJ à partir de message externe
Alors 1- envoi message_ interne =<<MAJ effectuée>>
au module communication ;

2-Allez à : etiq13 ;
Sinon
1- envoi message_ interne =<<MAJ effectuée>>au module de décision et planification des tâches.
2-Allez à : etiq26 ;

Fin de Si ;

Etiq16: 1- sauvegarde des assignations des fragment aux agents fragments

2-Etablir les facteurs d'évaluations du modèle de procédé modélisé

3-envoi message_ interne =<<modèle de procédé logiciel globale >>
Au module gestionnaire de la base des connaissances.

4-Allez à : **Etiq1.**

Etiq3: 1-Sélictionné la partie de modèle a exécuter.

2- envoi message_ interne =<< partie de modèle a exécuter>>
Au module de décision et de planification des tâches

3-Allez à : **etiq4.**

Etiq7: 1-introduire les changements dans les modèles de procédé globale.

2- envoi message_ interne =<<changement de modèle de procédé effectué>>
Au Module gestionnaire de la base des connaissances.

3-Allez à : **etiq8**

Etiq10: 1- introduire les changements dans les modèles de procédé globale en interaction avec le chef de projet.

2- envoi message_ interne =<<changement de modèle de procédé effectué>>
Au Module gestionnaire de la base des connaissances.

3-Allez à : **Etiq8**

Etiq23: Si fin d'exécution de tout le modèle de procédé
Alors : 1- envoi message_ interne =<<Envoi le résultat final de l'exécution>>
au module gestionnaire de la base de connaissances.

2-Allez à : **etiq24 :**

Sinon Allez à : **etiq3 ;**

Fin de Si;

2- envoi message_ interne =<< assignations des fragment effectuée >>
Au Module réalisation et évaluation de la réalisation.

3-Allez à : **etiq17** ;

Etiq19: 1- sauvegarde du rapport d'exécutions.

2- envoi message_ interne =<< rapport d'exécutions effectuée>>
Au Module communication

3- Allez à :**Etiq20** ;

Etiq24: 1- envoi message_ interne =<<Résultat final ou partiel d'exécution>>
Au Module communication

2-Allez à :**Etiq25** ;

Fin ;

- **Module de décision et de planification des taches tâches.**

Début :

Etiq4: 1-Fragmenter le modèle de procédé logiciels.

2-Etablir les facture des évaluations de chaque fragment.

3-Envoi message_ interne modèle de procédé prêt à s'exécuter
Au module de Réalisation et évaluation de la réalisation.

4-Allez à : **etiq25** ;

Etiq14: 1-consulter la stratégie utilisée

2-vérifie l'état de ressources

3-décision selon les objectifs a suivre et les priorité

Si changements dans l'état de ressources au des accointances
Alors 1- envoi message_ interne=<< changements d'état de

ressources>>
Au gestionnaire des Accointances

2-Allez à : **etiq11** ;
Fin de Si ;

Etiq14: 1- envoi message_ interne=<<Repense message d'interaction >>
Au Module de communication

2-Allez à :**etiq15** ;

Fin ;

❖ Module réalisation et évaluation de la réalisation

Début :
Etiq5: 1-Initialiser les agents fragments nécessaire.

2-Assigné les fragments et leurs facteurs d'évaluation aux agents correspondant

3- Envoi message_ interne=<<assignation des fragments aux agents fragment>>
Au Module gestionnaire de la base de connaissances

4-Allez à : **etiq16** ;

Etiq17: 1-Envoi message_ interne=<< assignation des fragments et facteurs d'évaluation>>
Au module de communication

2-Allez à : **Etiq6** ;

Etiq1: 1- envoi message_ interne=<< assignation des fragments et facteurs D'évaluation>>au module gestionnaire du modèle de procédé logiciel

2-Allez à : **Etiq3**;

Etiq21: 1-Evaluation des rapports d'exécutions

Si déviation du modèle instancier

Alors

1-consultation de module des ressources;

2-consultation des facteurs d'évaluations;

3-consultation des priorités d'exécution;

4-proposition des changements dans le modèle de procédé logiciel;

5- envoi message_ interne=<< proposition des changements dans le modèle de procédé logiciel>>
Au module de communication;

6-Allez à : **etiq22;**

Sinon

Si dernier rapport d'exécution de la partie assignée
Alors:

1-envoi message_ interne=<<fin d'exécution de la partie assignée>>
Au Module gestionnaire de procédé logiciel

2- Allez à : **etiq23;**

Fin de Sinon ;

Fin de Si

Fin.

❖ Module gestionnaire des accointances et de ressources

Début :

Etiq17: 1- envoi message_ interne=<<Etat es ressources ou des accointances mis a jours>>.
Au module gestionnaire de base de connaissances

2- Allez à : **etiq23;**

Fin ;

Conclusion :

Dans ce chapitre nous avons présenté l'architecture interne de notre agent superviseur, de plus, nous avons essayée de bien expliquer la solution proposer par les algorithmes définis dans cette partie qui sont écrits de manière globale ; plusieurs points restent à développer, tel que les stratégies de négociation utilisées, le choix des stratégies, la manière d'introduire les changements....etc. Nous n'avons pas approfondi ces points de manière délivrée, car notre objectif principal est de définir la structure générale des algorithmes.

Après la présentation d'une architecture de la solution proposée, en passe maintenant à la conception de notre agent.

CHAPITRE IV:

Conception du modèle de procédé logiciel

Introduction

Pour concevoir notre système nous avons utilisé les diagrammes UML afin de ressortir l'architecture du système, dont les notations graphiques des différents diagrammes respectent les spécifications UML.

Le choix du langage de modélisation porte sur le langage UML de part le fait qu'il constitue une unification des méthodes objets tirant donc profit des avantages de chacune de ces méthodes de l'origine de l'unification et qu'il constitue un standard pour la modélisation orientés objet.

Ce choix est également justifie par plusieurs autres avantages qu'offre UML par rapport a d'autres méthodes objets et par rapport a nos besoins, nous les décrivons dans ce qui suit :

1- UML est fondé sur un méta modèle donne une vision plus formalisée du langage et définit toutes les possibilités d'extensibilité de celui-ci.

2- UML se veut un langage non fermé: les éléments de modélisation qu'il propose sont génériques, extensibles et configurables par l'utilisateur

3- La simplicité et l'expression visuelle qu'offre l'ULM, et qui permettent de faciliter la communication autour des besoins entre les différents acteurs du système. UML offre une bonne communication avec les utilisateurs. Les concepts manipulés en UML correspondent à des réalités concrètes pour l'utilisateur.

4- La diversité des diagrammes qu'offre UML et qui permet de présenter plusieurs perspectives ou vues de l'architecture du système a développer.

On a choisi de suivre la démarche RUP ainsi on a utilisé les diagramme suivants:

-Diagramme de cas d'utilisation.
-Diagramme des composants.
-Diagramme de classe.
-Diagramme de séquence.
-Diagramme de collaboration.
-Diagramme d'activité.
-Diagramme d'état/transition.

IV-1-Diagramme de cas d'utilisation :

Les diagrammes des cas d'utilisations présentent les cas d'utilisations, les acteurs et les relations entre les cas d'utilisations et les acteurs. Un cas d'utilisation est une manière de décrire comment utiliser le système .c'est l'image d'une fonctionnalité du système déclenché par un acteur.**[11]**

IV-1-1-Les acteurs:

Se sont les catégories d'utilisateur (humain ou systématique) qui interagit avec notre système (les acteurs sont à l'extérieur du système).

Les acteurs de notre système sont les suivants :
a- AGENT SUPERVISEUR.
b- AGENT FRAGMENT.
c- CHEF DE PROJET.

IV-1-2-Déterminations des cas d'utilisation :

a- Suivre la modélisation des procédés logiciels :

Le chef de projet réalise cette tache car la modélisation de procédés logiciels ne peut pas être effectué par l'agent superviseur seul, et l'intervention du chef de projet reste fondamentale pendent tout les étapes de la modélisation des procédés logiciels le chef de projet et l'agent superviseur doivent suivre la modélisation étape par étape pour confirmer ou stopper une étape de modélisation a n'import quelle moment.
"Voir chapitre 2 pour plus de détails".

b- Etablir les facteurs d'évaluation :

Cela se fait par l'agent superviseur et peut être modifié par le chef de projet.
"Voir chapitre 2 pour plus de détails".

c- Stock le modèle de procédés dans la base de connaissance :

Lorsque l'agent superviseur termine la modélisation de procède logiciels ; il doit stocker ce dernier dans sa base de connaissance. "Voir chapitre 2 pour plus de détails".

d- La fragmentation :

La fragmentation consiste a diviser le procédé en plusieurs parties, phase ou fragments. "Voir chapitre 2 pour plus de détails".

e- Introduire des changements dans le modèle de procédés logiciel :

Le chef de projet peut aussi introduire des changements dans le modèle de procède logiciels après sont début d'exécution et donne des consignes de modélisation en interaction avec l'agent superviseur.

f- Assignation des fragments procédés logiciels aux agents fragments

Cela consiste à créer un agent fragment et lui assigné un fragment pour l'exécuter.

Voici le diagramme de cas d'utilisation

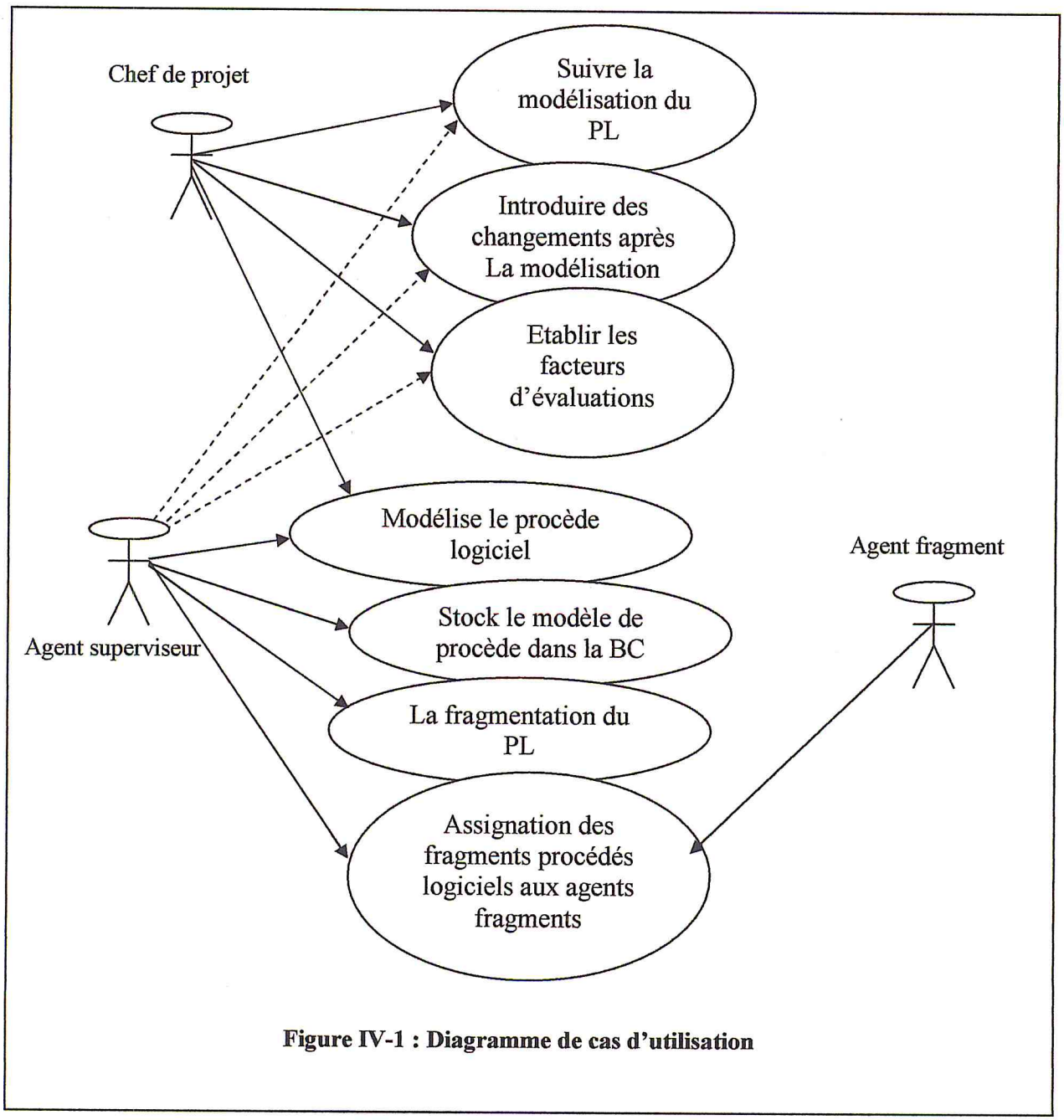


Figure IV-1 : Diagramme de cas d'utilisation

IV-2-Diagramme des composant.

Les diagrammes de composants permettent de décrire l'architecture physique et statique d'une application en terme de modules. Ils montrent la mise en oeuvre physique des modèles de la vue logique avec l'environnement de développement.

Les dépendances entre composants permettent notamment d'identifier les contraintes de compilation et de mettre en évidence la réutilisation de composants.

Les composants peuvent être organisés en paquetages, qui définissent des sous-systèmes. Les sous-systèmes organisent la vue des composants (de réalisation) d'un système. Ils permettent de gérer la complexité, par encapsulation des détails d'implémentation.

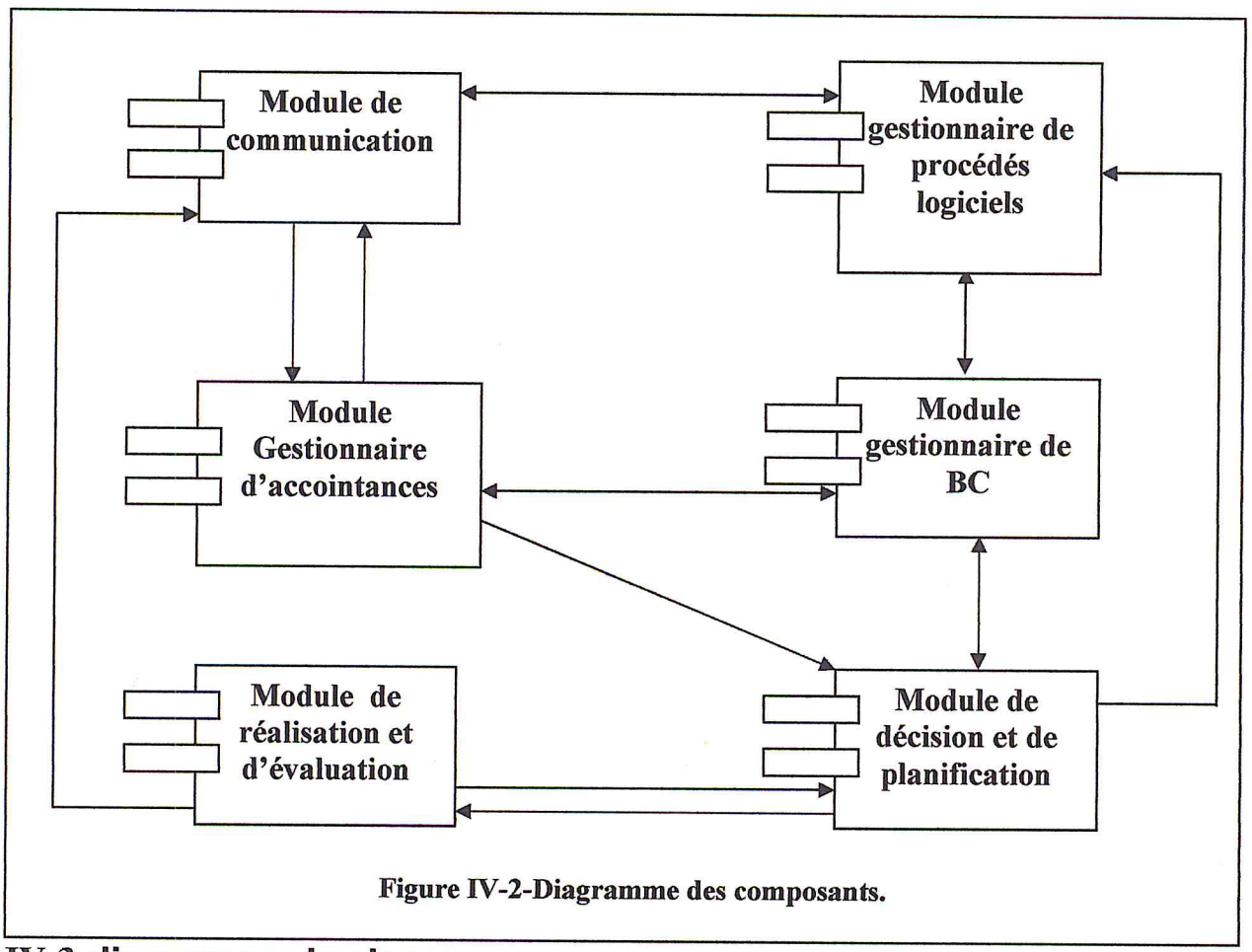


Figure IV-2-Diagramme des composants.

IV-3-diagramme de classe :

Le diagramme de classe exprime de manière générale la structure statique d'un système en terme de classe et de relations entre ces classes **[3]** .le diagramme de classe suivant représente les éléments qui doivent être

implémentés dans notre système, ces éléments représentent les connaissances de l'agent.

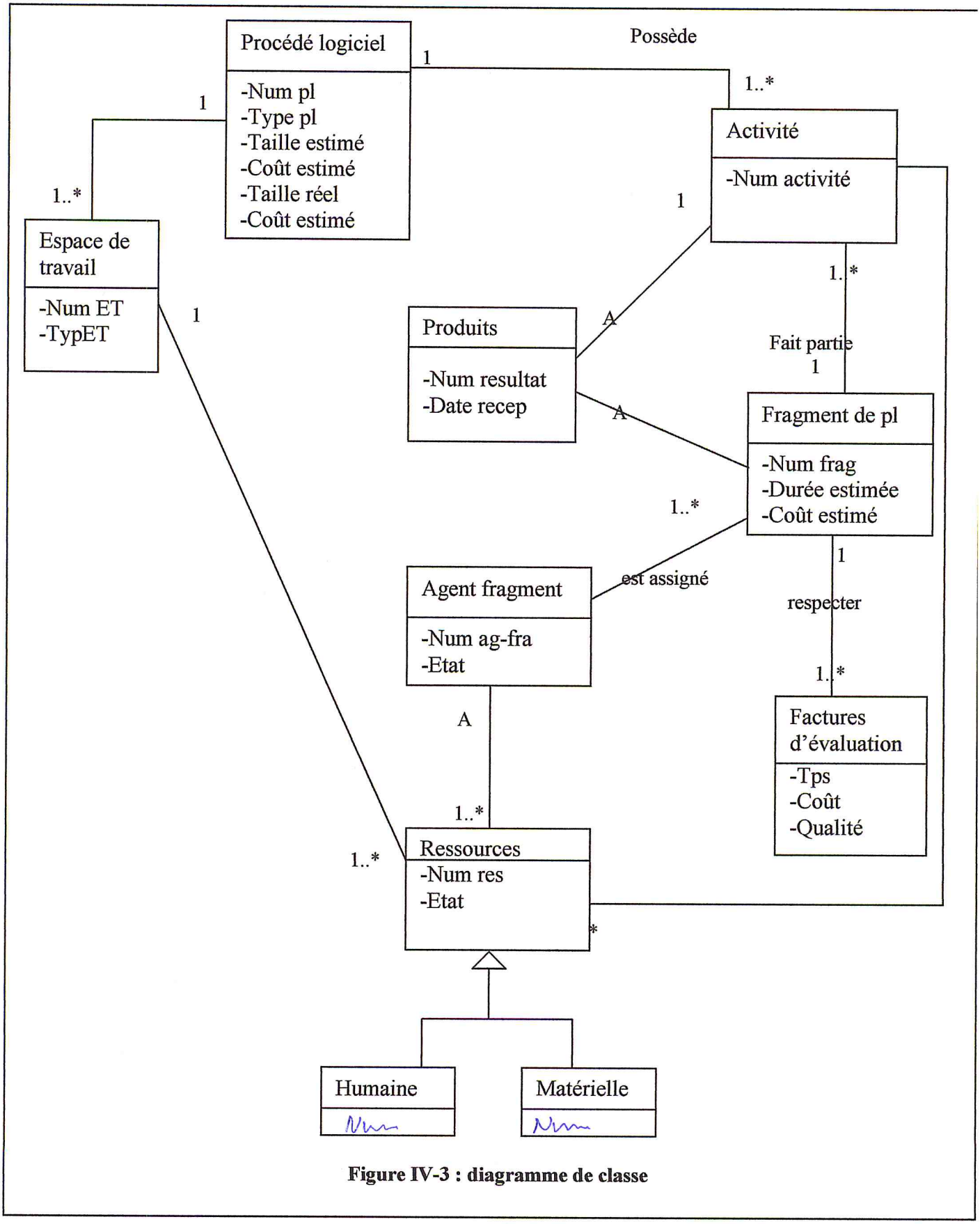


Figure IV-3 : diagramme de classe

IV-4-Diagramme de séquence :

Les diagrammes de séquence nous permettent de montrer les interactions entre objets, ils nous permettent de bien schématiser les scénarios des cas d'utilisation.

La représentation se compte sur la séquence des interactions selon un points de vue temporel ils sont en générale plus aptes à modéliser les aspects dynamiques des systèmes temps réel et des scénarios complexes mettent en œuvre peu d'objets. **[11]**

Une interaction modélise un comportement dynamique entre objets. Elle se traduit par l'envoi de message entre objets.

Un diagramme de séquence représente une interaction entre objet en insistent sur la chronologie des envois des message. **[11]**

Un objet est matérialisé par un rectangle ou une barre verticale, appelée ligne de vie de l'objet **[11]**, cette barre verticale représente l'écoulement de temps (de haut en bas) .les objets communiquent en échangeant des messages représentés au moyen des flèches horizontales, orientées de l'émetteur du message vers le destinataire .l'ordre d'envoi des messages est ainsi donné par la position de ces messages sur les lignes de vie de objets.

IV-4-1-Diagramme de séquence globale :

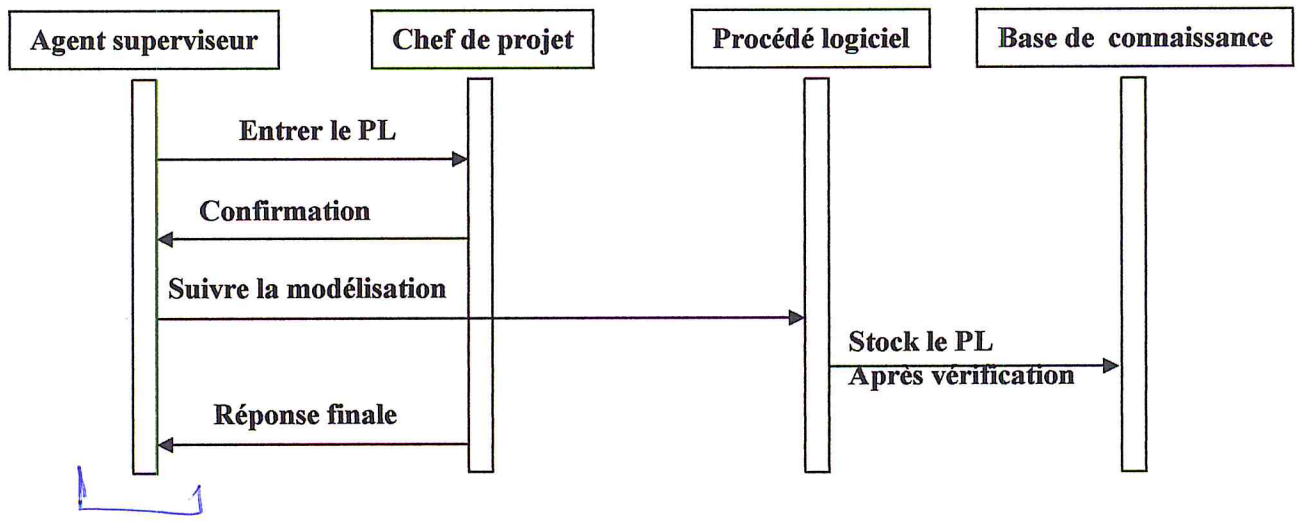


Figure IV-4 : diagramme de séquences globale

IV-4-2-Diagramme de séquence de cas d'utilisation de la fragmentation du PL et assignations et établissements des facteurs d'évaluation:

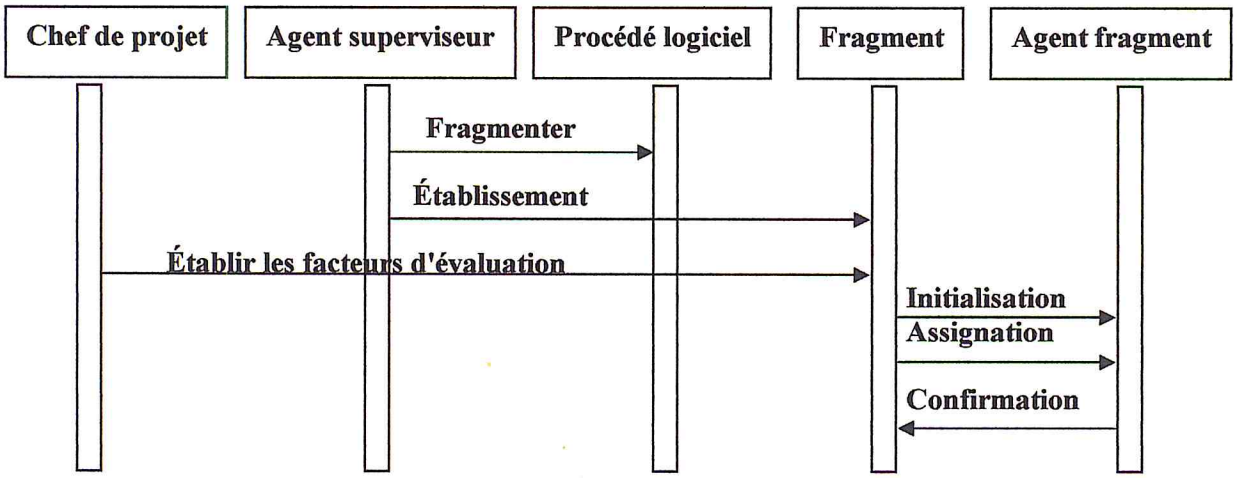


Figure IV-5 : diagramme de séquence de cas d'utilisation de la fragmentation du PL et d'assignations et d'établissements des facteurs d'évaluation.

IV-4-3-Diagramme de séquence de cas d'utilisation d'introduction des changements dans le modèle de PL:

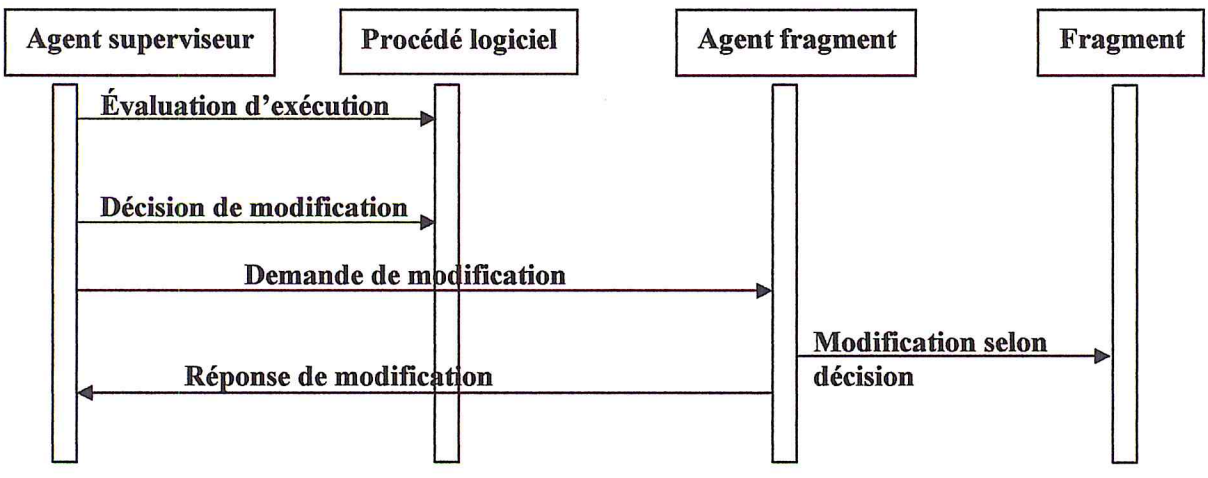


Figure IV-6 : diagramme de séquence de cas d'utilisation d'introduction des changements dans le modèle de PL

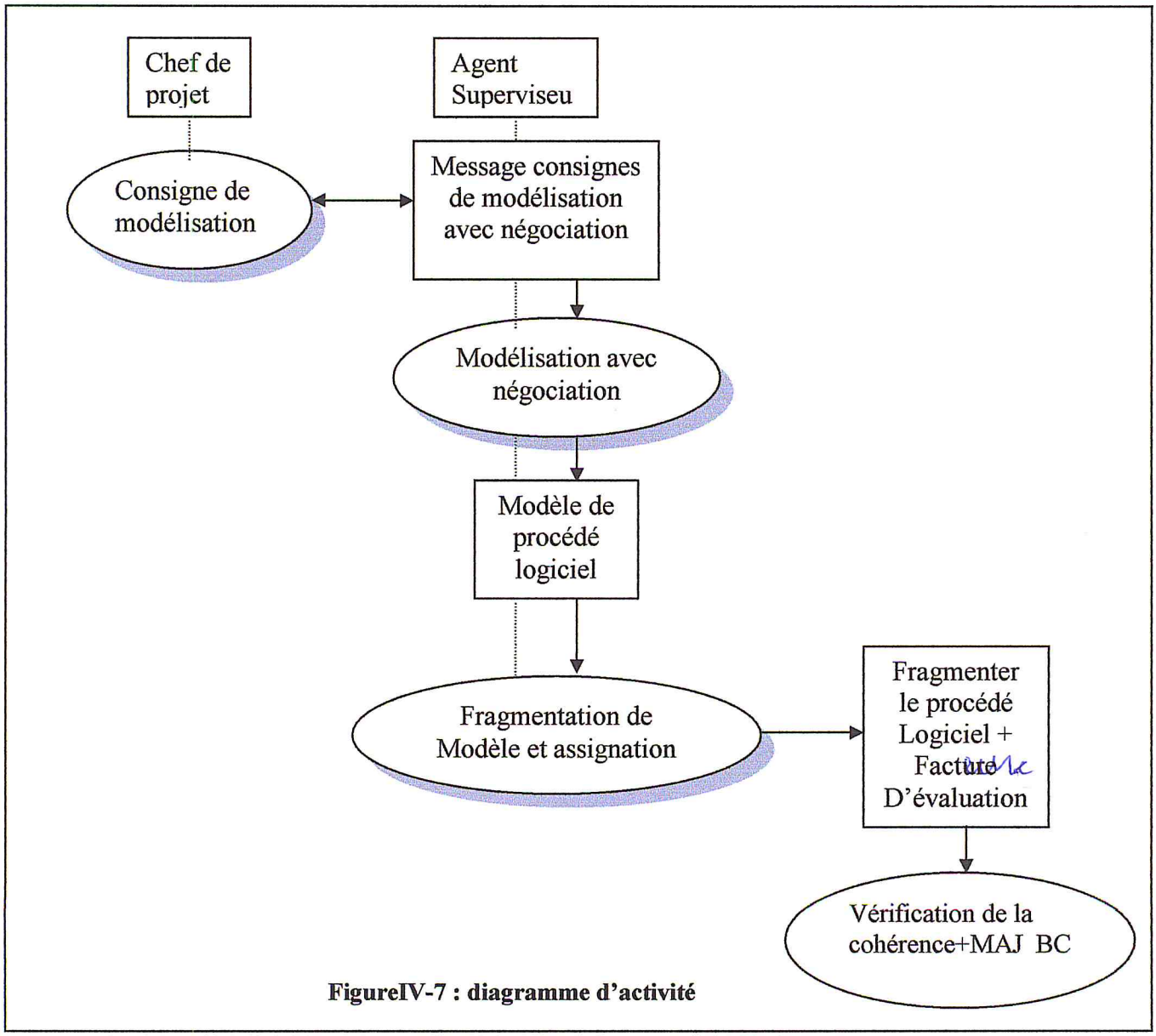
VI-5-Diagramme d'activité :

UML permet de représenter graphiquement le comportement d'une méthode ou le déroulement d'un mécanisme à l'aide d'un diagramme d'activité.

Le diagramme d'activité permet de représenter les activités internes de l'agent modèle de procédé et les interactions essentielles avec ces voisins.

Les couloires d'activités représentent dans la plupart des cas les voisins de l'agent qui peuvent être un autre agent ou le chef de projet.

Sachant que chaque type d'agent a des objectifs qui régissent son comportement est des voisins distincts, il est naturel d'avoir des diagrammes d'activités qui varie d'un type d'agent à un autre ,Ainsi les couloirs d'activités de l'agent superviseur est le chef de projet.



FigureIV-7 : diagramme d'activité

IV-6-Diagramme de Collaborations :

Le diagramme de collaboration représente une vue dynamique du système. Il montre également les interactions entre les objets à travers la représentation d'envoi de messages. L'ajout d'une dimension temporelle requiert la définition de numéro de séquence pour les messages.
Une collaboration définit les éléments qui sont utiles pour l'obtention d'un objectif particulier en spécifient le rôle de ces éléments dans un contexte de la collaboration **[11]**.

Le premier message de l'interaction est envoyé par l'acteur, un message se présente par une flèche placer a proximité d'un lien et dirigée vers l'objet destinataire du message au sein de l'interaction. En générale, une séquence dans un diagramme de collaboration commence par le chiffre 1. Les numéros de séquence suivants sont incrémentés.
Voici le diagramme de collaboration de notre système: (page suivante)

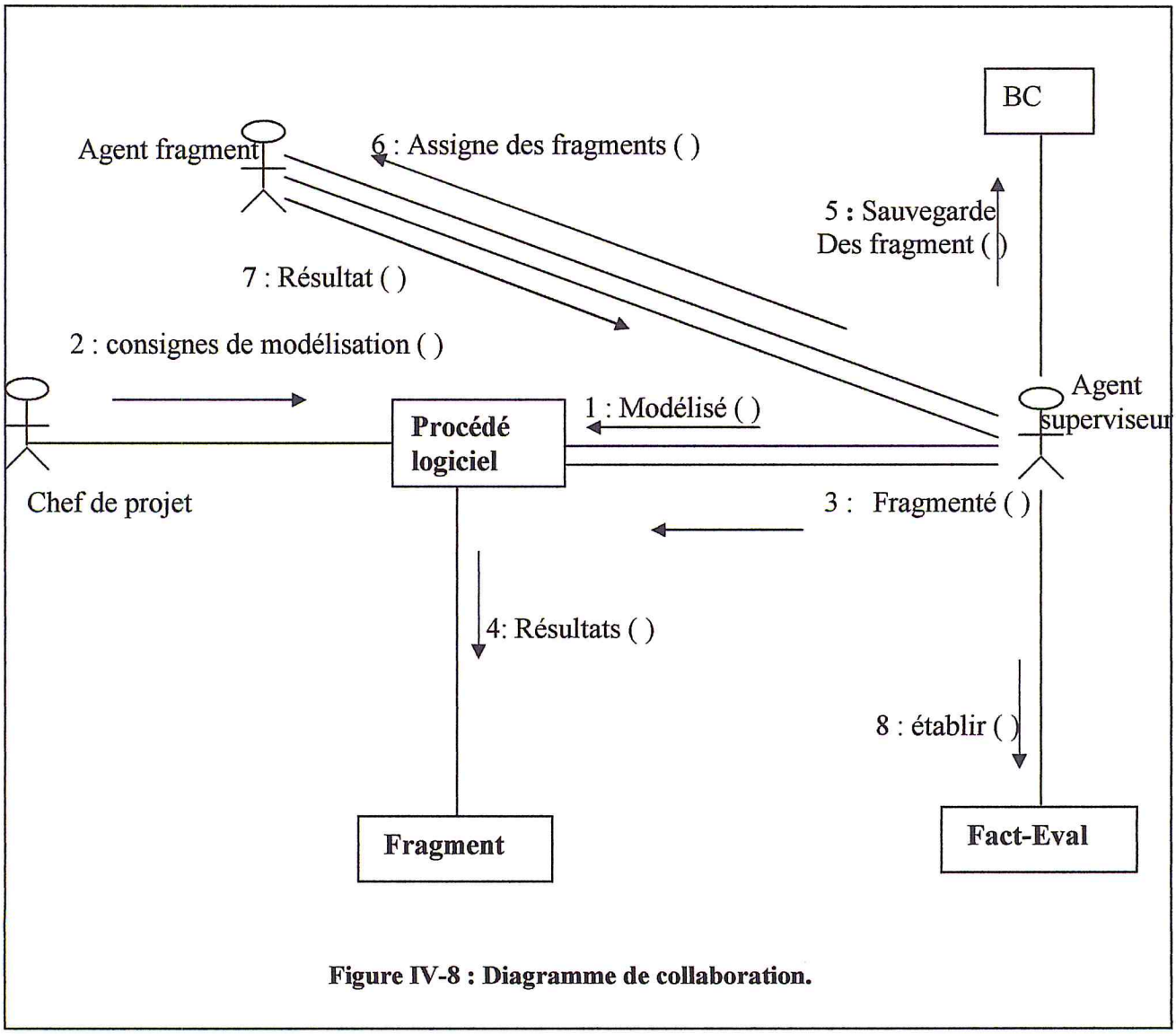


Figure IV-8 : Diagramme de collaboration.

VI-7-Diagramme d'Etat /transition :

Le diagramme d'états/transitions à pour objectif de représenter des traitements permettant de gérer les domaines étudies. Ces traitements (opérations) sont positionnés par apport à des classes et plus précisément par apport à des états de classes. Le diagramme mettant en évidence l'enchaînement des différents états d'une classe fait ainsi apparaître l'ordonnancement de ces déférents travaux.

Le diagramme d'états transitions est associe a une classe pour laquelle on gère ces déférents états, il permet de représenter tout les états possibles, ainsi que tout les évènements provoquant un changement d'état.

VII-7-1- Diagramme d'Etat /transition Agent superviseur :

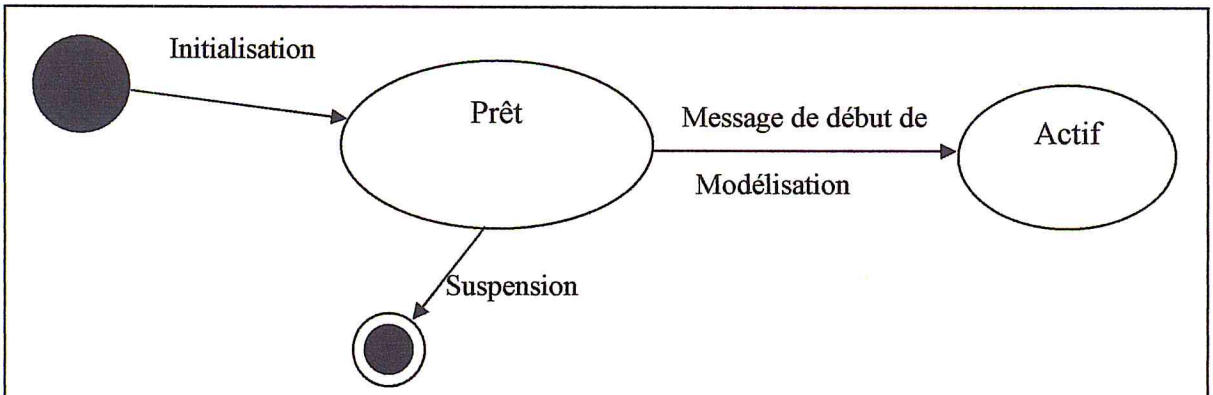
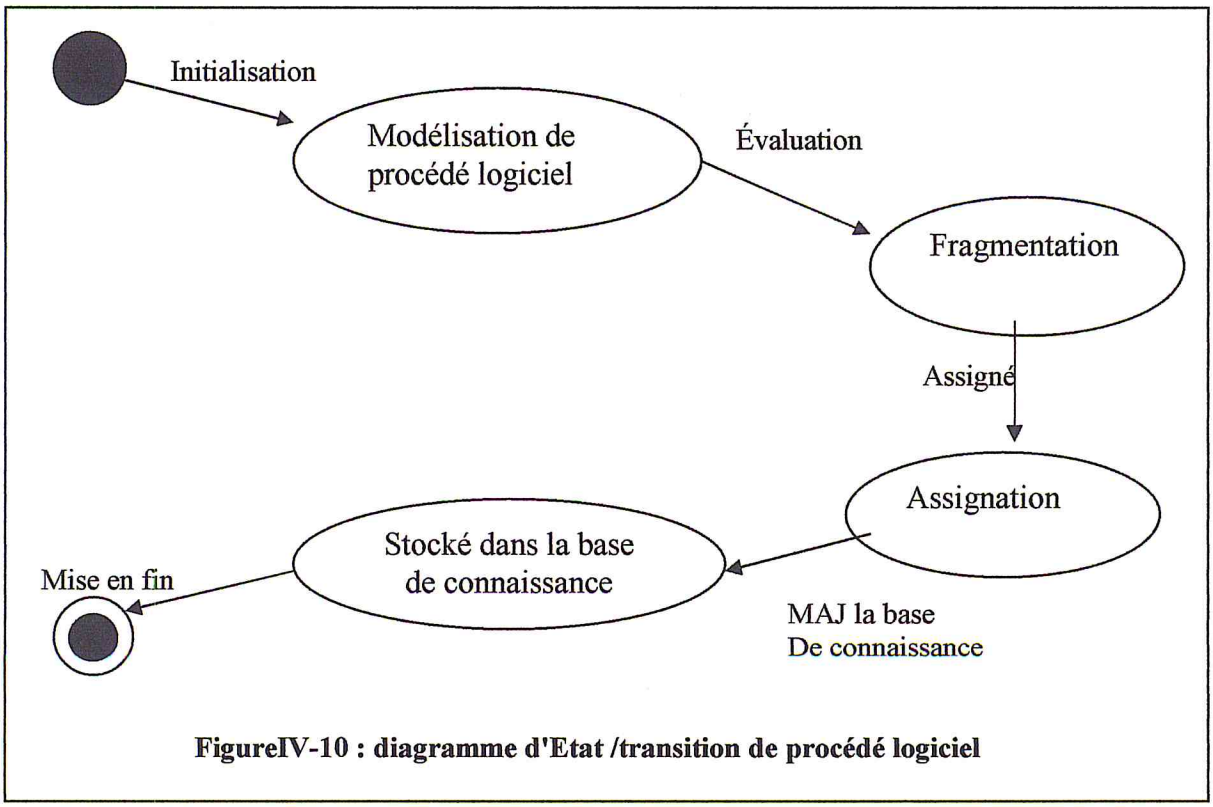


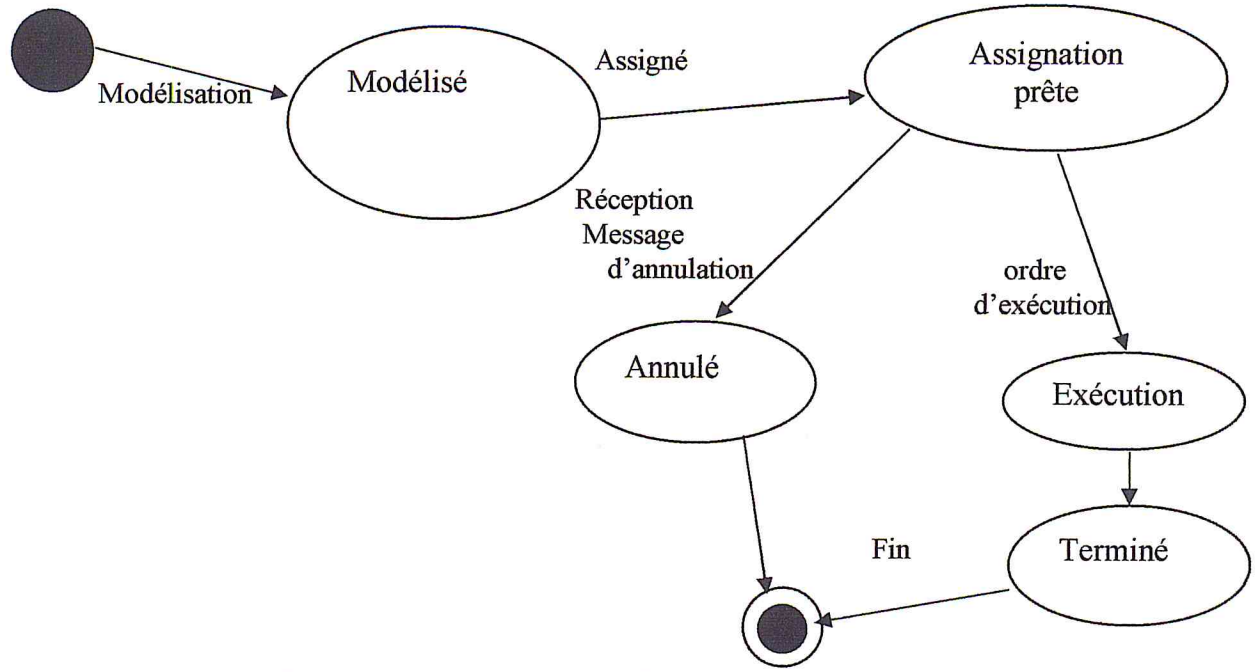
Figure-IV-9 : diagramme de Etat /transition de L'agent superviseur

VII-7-2- Diagramme d'Etat /transition Procédé logiciel :



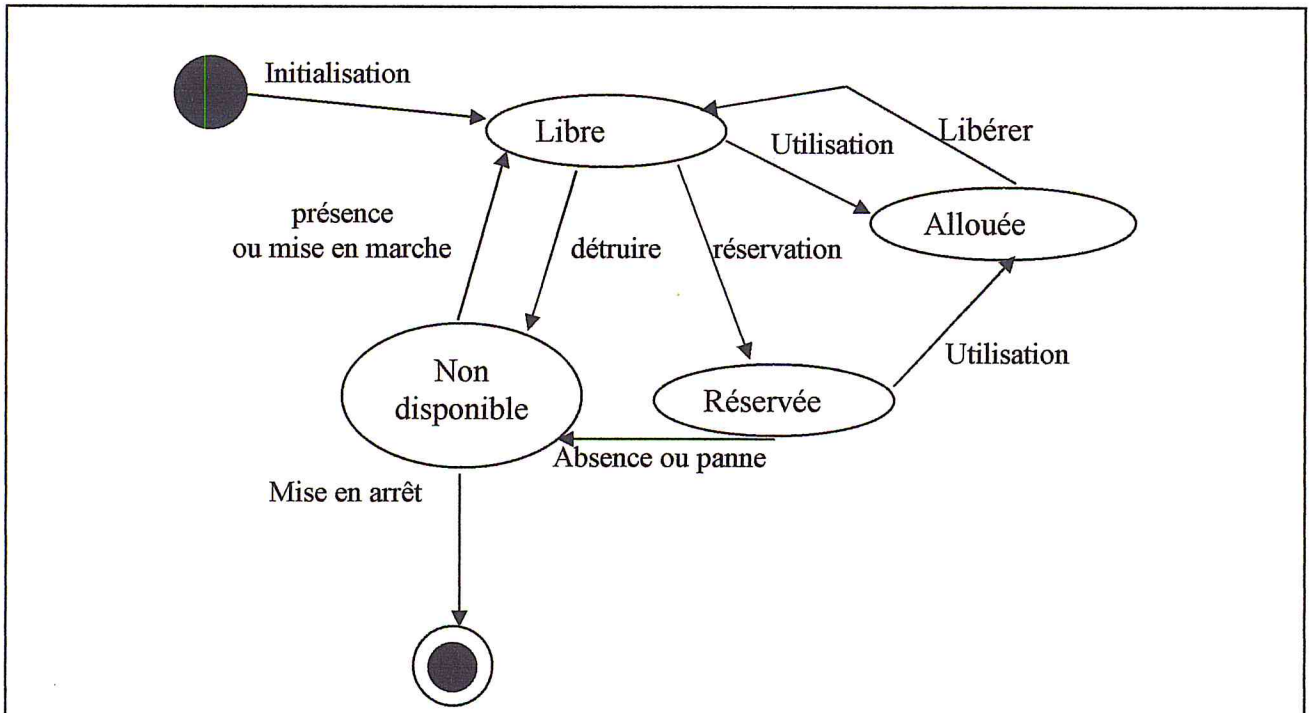
FigureIV-10 : diagramme d'Etat /transition de procédé logiciel

VII-7-3-Diagramme d'Etat /transition activité:



FigureIV-11 : diagramme d'Etat /transition activité

VII-7-4-Diagramme d'Etat /transition ressource:



FigureIV-12 : diagramme d'Etat /transition ressource

Conclusion :

Dans ce chapitre, nous avons présenté une conception de l'application et une conception du système agent modèle de procédé logiciel, de plus par l'intermédiaire de plusieurs diagrammes, nous avons essayé de bien expliquer la solution proposer tout en illustrons l'ensemble des ses composants et leur manière d'agir, après la modélisation de la solution proposée, en passe maintenant à la réalisation, ce qui est proposé dans le chapitre suivant.

CHAPITRE V
Implémentation et mise en œuvre

Introduction

Après l'architecture et la conception de notre agent superviseur dans les chapitres précédents, nous allons présenter dans ce chapitre une proposition d'implémentation et de mise en œuvre de notre agent superviseur en commençant par la description de l'implémentation et les outil de développement a utilisés et après on détaillera la mise en œuvre et les démarches à suivre pour arriver à modéliser un projet.

V-1- Les outils de développement:

V-1-1- Plateforme génératrice d'agent: (Les Aglets)

Pour implémenter l'agent on a choisis "Aglet" comme plateforme de génération d'agents, citons que Aglet est une plateforme basée sur Java pour développer des agents dans un environnement hétérogène.

En 1996 Big Blue (IBM) met à disposition " Aglets Worbench " qui regroupe une librairie sous forme d'un package Java, ainsi qu'un serveur d'aglets (Tahiti). " Aglets Workbench " sera renommé en 1998, ASDK (Aglets Software Developement Kit). La ASDK est destiné à faciliter la création d'Agent intelligent mobile.

Aglets vient des mots "Agent" et "Applet". Quand nous parlons du serveur d'Aglets, ceci est équivalent à un serveur d'Agent pour ce système.

Aglet Java est une norme proposée pour développer des Agents intelligent et mobiles basé sur java. Aglet a été développé par une équipe de chercheurs du laboratoire de recherche d'IBM Tokyo au Japon; son but est de fournir une plate-forme uniforme pour les Agents mobiles dans un environnement hétérogène tel que celui de l'internent.

Aglet permet d'écrire en une seule fois les applications, qui s'exécuteront partout avec les avantages liés à Java. On n'aura plus à se soucier de la couche matérielle ou du système d'exploitation, ni de la manière dont le code de l'aglet a été programmé.

Le produit ASDK n'est pas encore mûr pour une utilisation en production. Nous attendons les outils d'administration pour des déploiements à grande échelle, Mais il reste un produit à surveiller.

Ces messages sont utilisés pour la communication entre l'agent superviseur et l'agent fragment.

Pour plus de détaille sur les aglets vous consultez les annexes.

V-1-2- Langage de construction de l'interface d'interaction Chef de projet/Agent superviseur : (C++ Builder)

Pour construire l'interface de modélisation des procédés logiciels qui sera l'interface d'interaction entre le chef de projet et l'agent superviseur et pour effectuer les calcules des facteurs d'évaluation on a choisis le langage de programmation "C++ Builder", parce qu'il est très simple et facile à la construction des interfaces d'interaction avec ce langage et aussi parce qu'on peut faire des appels à un programme C++ Builder à partir d'un programme java ou bien Aglet.

Borland c++ builder est un langage de développement rapide (RAD: rapide application développement), Avec C++ builder, Borland introduit une autre révolution pour les programmeurs C++, la programmation d'application Windows s'en trouve considérablement Simplifier pour ces développeurs qui contentent cependant à bénéficient de la puissance du C++ et des techniques de programmations orienté objet, Ajour d'Hui ,Avec C++ builder ,on peut dire qu'ils bénéficient à la fois de la puissance et de la facilité de développement. C++ builder rend en effet le développement d'une application Windows incontestablement plus rapide qu'avec les outils C++ disponible jusqu'à ce jour.

Développer une application Windows à l'aide de C++ builder Consiste en effet à:

- Concevoir l'interface utilisateur de manière particulièrement visuel et interactif.

- Spécifier de manière tout aussi interactive les caractéristiques appelées propriétés, des divers composants (fenêtres, boutons de commande, zones d'édition, etc....).

- Spécifier le code à exécuter lorsque l'utilisateur effectué telle ou telle action. **[10].**

Voici le code de base d'un aglet:

```
Public class agent extends Aglet {
Les variables utilises (public, private, protected)
.....
...
    }
Public void on Création () {
//Code a exécuté par l'agent superviseur
//c'est dans cette classe qu'on va appeler l'interface de communication.
.....
....
    }
Public void run(){
// Code à exécuter par l'agent superviseur dans chaque environnement.
// Ici on peut entrer le code de l'agent fragment.
.....
                }
    }
```

Pour crée et envoyer un message entre les aglets, on utilise la fonction sendMessage .

```
Message msg = new Message ("bonjour");
someProxy.sendMessage (msg);
```

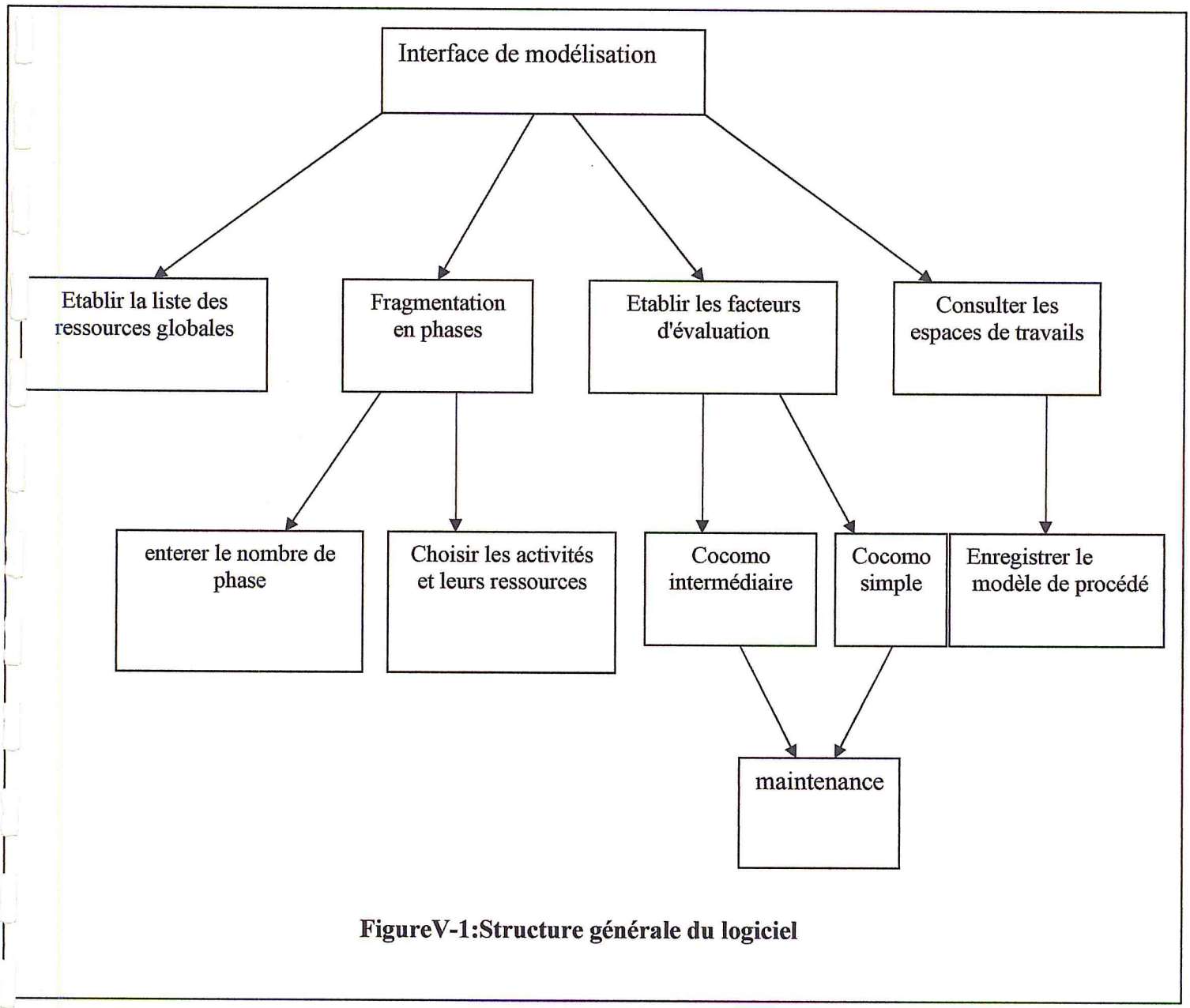
Pour pouvoir gérer le message reçu du type "message" chez le destinataire, on utilise la fonction handleMessage.

```
Public Boolean handle Message (Message msg)
    {
        if ("bonjour".equals (msg.kind)) {
           dobonjour ();
           return true;
        } else
           return false;
     }
```

V-2- Structure générale du logiciel

Pour expliquer comment que notre application fonctionne on a choisi de reprendre ce qu'on a dit en ce qui concerne les principales fonctions de notre agent et montrer comment réaliser chaque fonction.

Avant ça voici la structure générale de notre agent:



FigureV-1:Structure générale du logiciel

Les principales fonctions de notre agent superviseur (chapitre3) sont:

V-2-a-Modélisation des procédés logiciels et la Fragmentation

Cette tache est faite par plusieurs interfaces d'interactions entre le chef de projet et l'agent superviseur et dés que le chef de projet lance un nouveau projet elle s'affiche la fenêtre principale de l'interface de modélisation c'est la figure5-1.

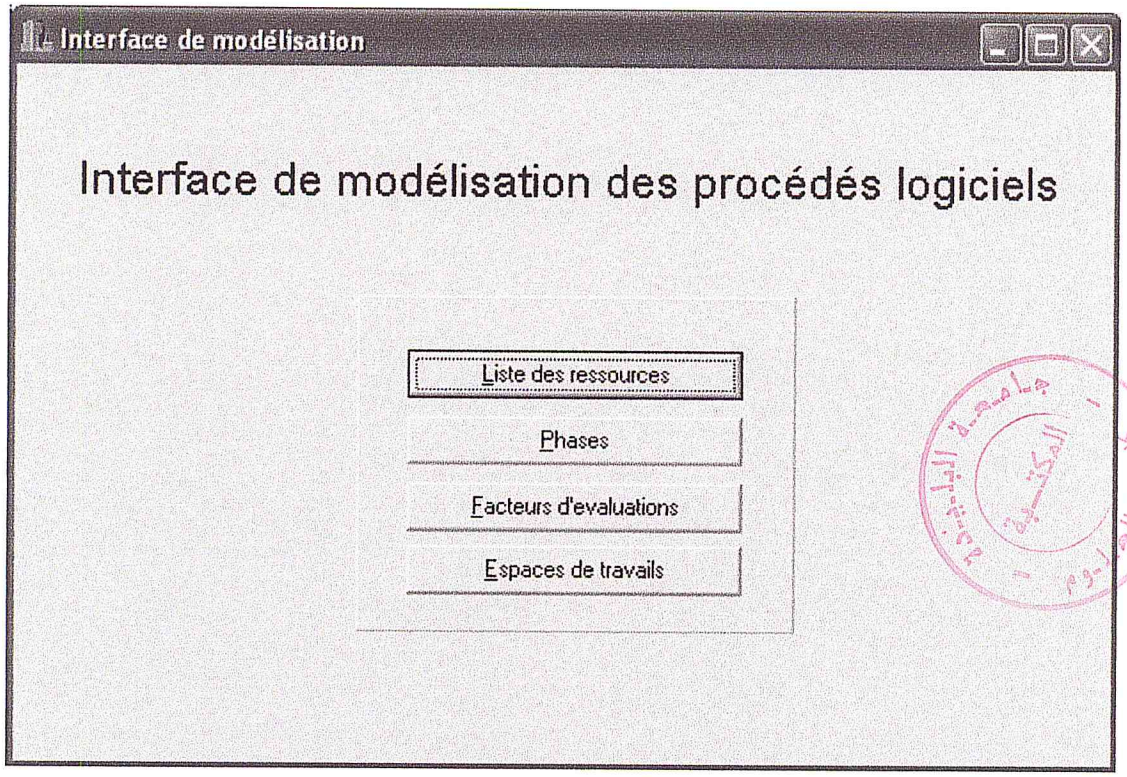
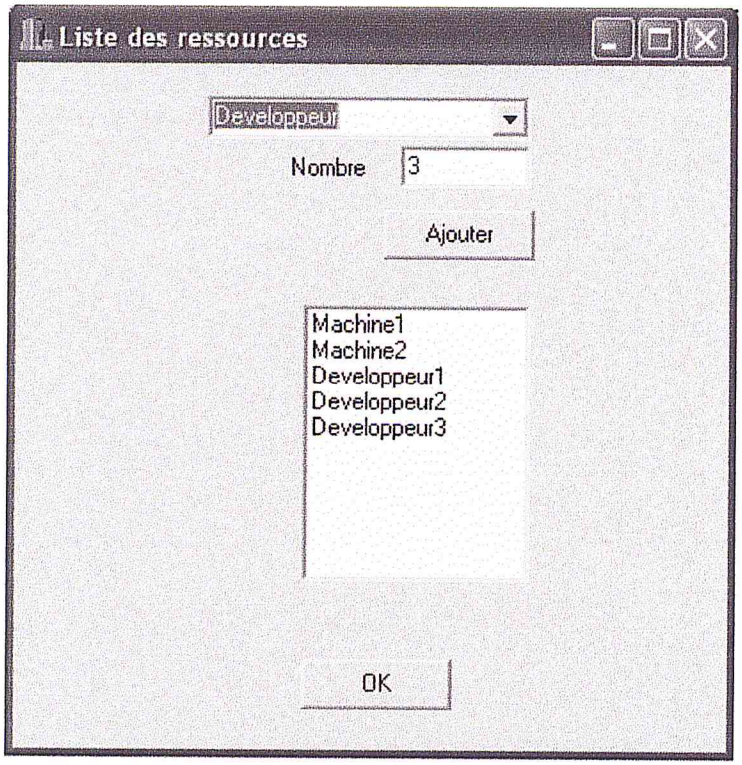


Figure V-2: interface de modélisation

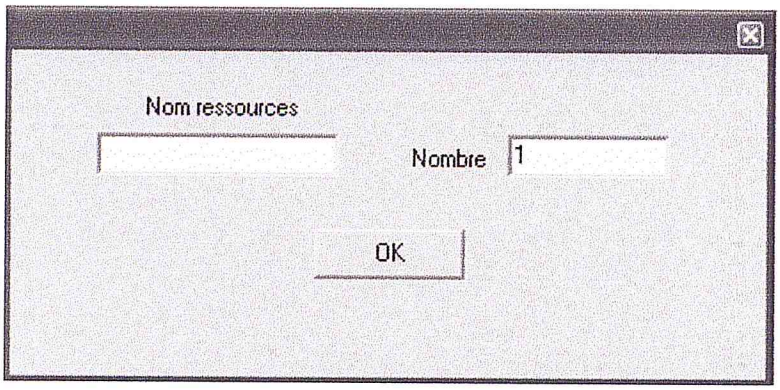
Cette fenêtre est composée de quatre boutons un pour choisir les ressources fournis à un projet l'autre pour construire nos phase "fragments" le troisième pour établir les facteurs d'évaluations et le dernier pour consulter notre architecture des espace de travail et tout ces boutons seront bien détaillé par la suite.

En cliquant sur "Liste des ressources" la fenêtre suivante s'affiche:Figure5-2



FigureV-3:Liste des ressources

Cette interface demande au chef de projet de sélectionner tout ce qu'il a comme ressources humaines et matérielles Qu'il veut réserver pour son projet. Dans la première case il va choisir dans une liste qui s'affiche le nom de la ressource, dans la deuxième case il doit entrer le nombre de ressources, et s'il ne trouve pas la ressource qu'il veut dans la liste il doit sélectionner "autres" pour choisir d'autres ressources dans la fenêtre suivante:

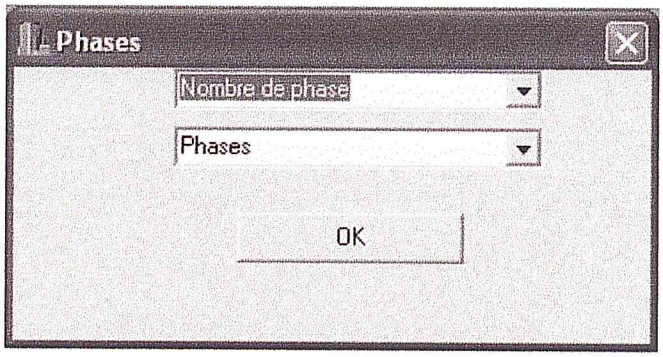


FigureV-4: Ajouter une nouvelle ressource

Dans la première case le chef de projet saisit la ressource qu'il veut et le nombre de ressources dans la deuxième case.

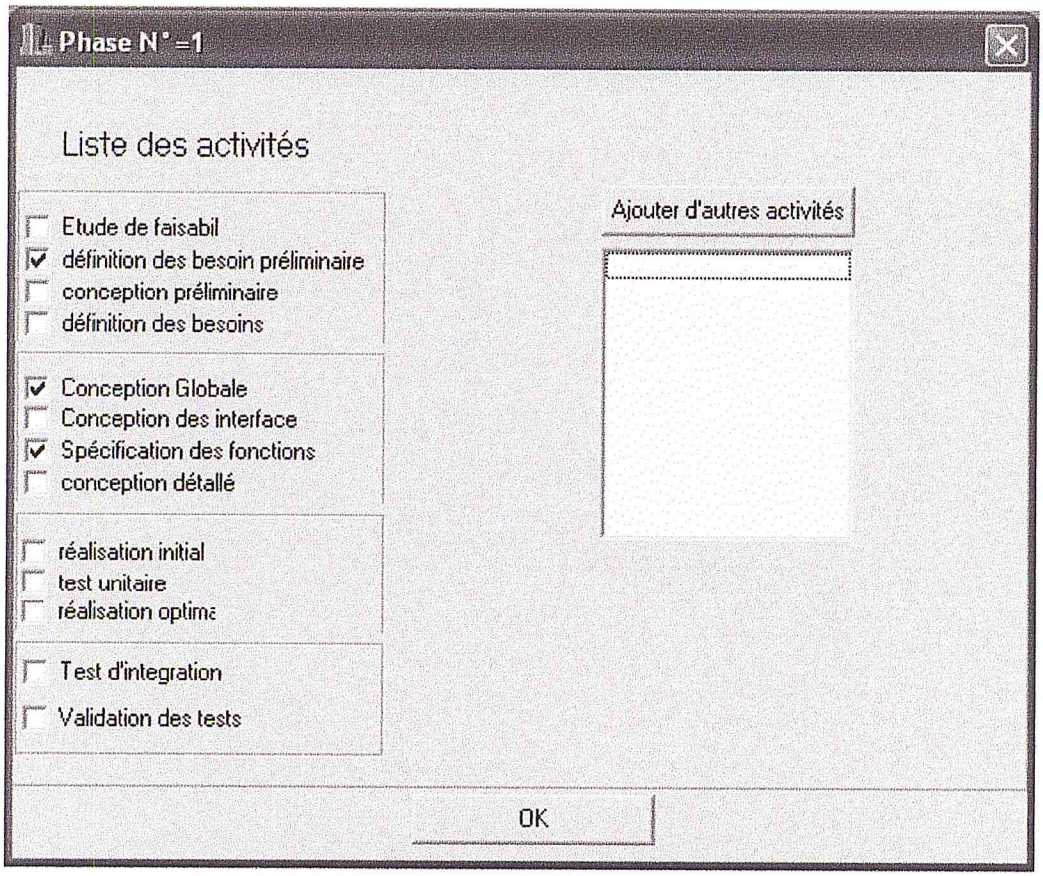
Lorsqu'il termine le choix de toutes les ressources il reviendra à la première fenêtre et il doit cliquer sur le bouton "Phase", l'interface qui s'affiche

demande au chef du projet de choisir en combien de parties "fragments" veut réaliser son projet.



FigureV-5: nombre de phases

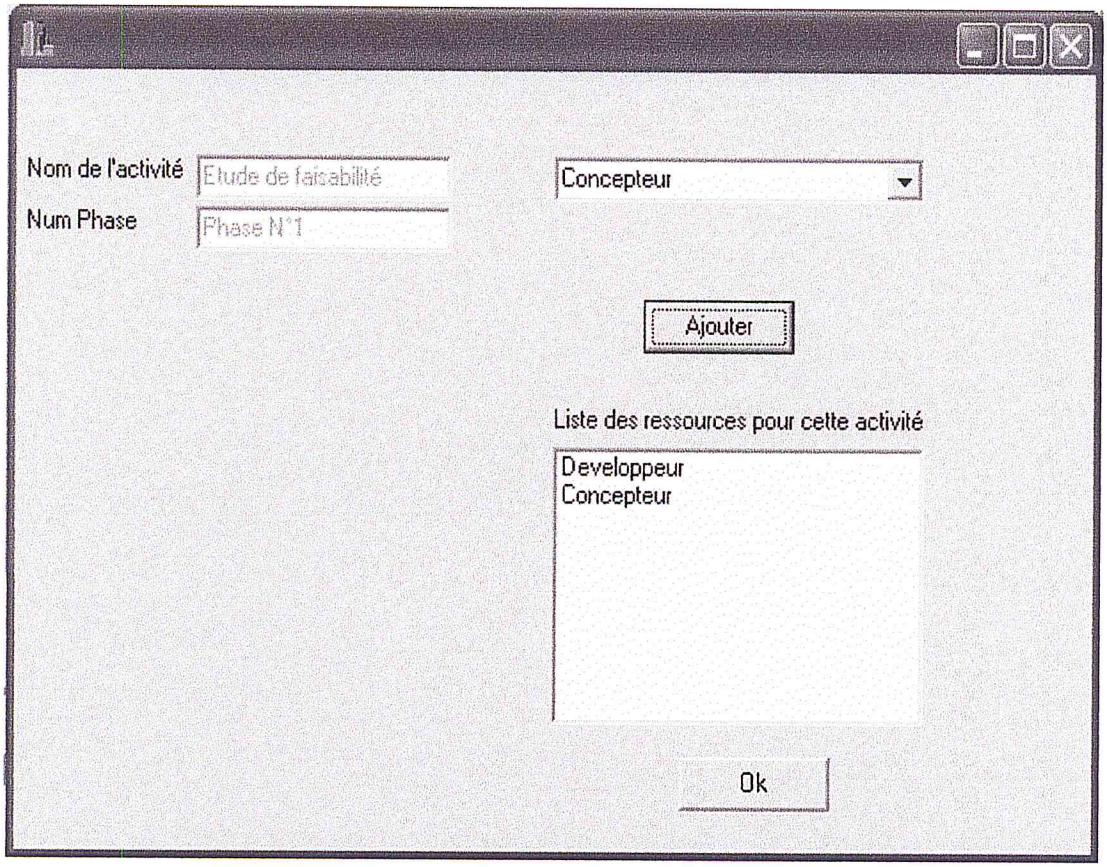
Ensuite il va choisir l'ensemble d'activités pour chaque fragment qu'il a choisit et cela se fait dans la fenêtre suivante.



FigureV-6:choix d'activités dans la phase N°1

Le choix d'activités se fait par la sélection d'activités dans la liste des activités ou en cliquant sur le bouton "Ajouter d'autres activité" pour ajouter d'autre activités qui ne se trouvent pas dans la liste des activité.

A chaque fois qu'il sélectionne une activité s'affiche une autre fenêtre qui lui demande de sélectionner ce qu'elle a besoins comme ressources, et cela se fait dans la fenêtre suivante.



FigureV-7: choix de ressources pour l'activité

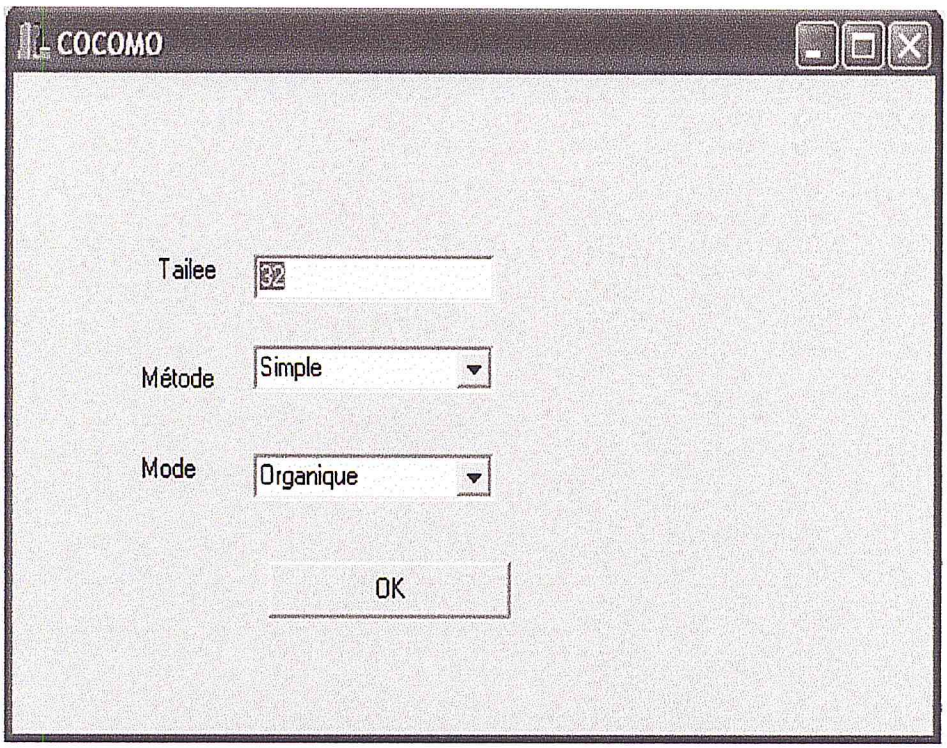
Le chef de projet choisit les ressources nécessaires pour chaque activité et cela se fait en sélectionnant la ressource dans la première case et en cliquant sur le bouton "Ajouter".

Après le partitionnement et le choix des activités ainsi leurs ressources on arrive à l'étape d'établissement des facteurs d'évaluation de tout le projet ainsi les facteurs de chaque phase.

V-2-b- Etablir les facteures d'evaluations:

Le calcule des facteurs d'évaluations se fait par la méthode *COCOMO* simple ou intermédiaire selon le choix du chef du projet.

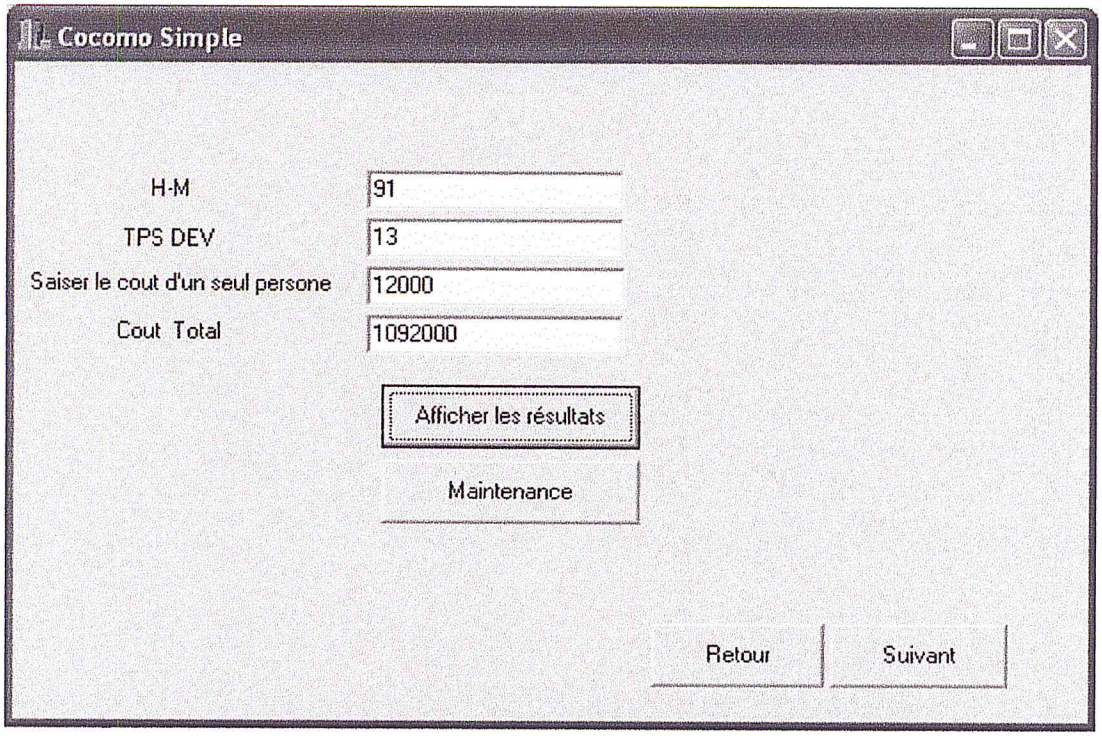
Pour cela le chef de projet revient à la première fenêtre et il clique sur le bouton "facteurs d'évaluations" pour aller à la fenêtre qui va l'aider à faire le calcule des facteurs et pour cela la première fenêtre qui s'affiche est la suivante:



FigureV-8: interface COCOMO

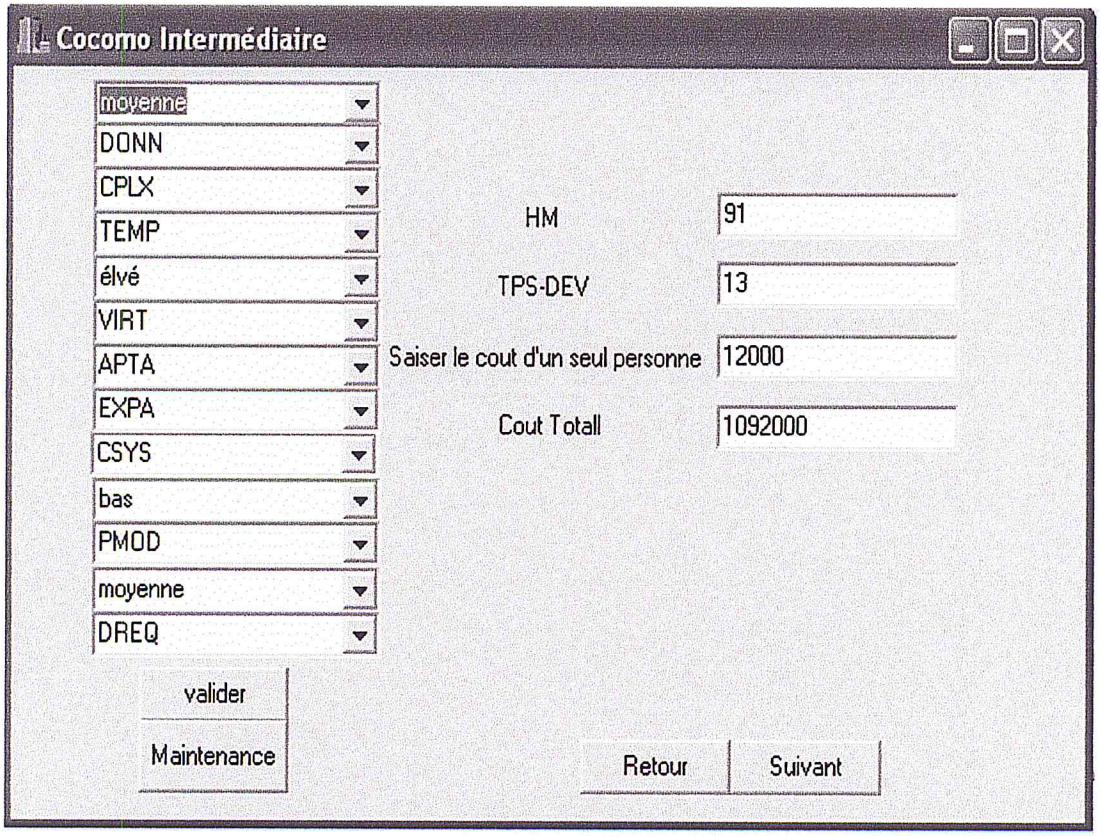
Dans la première case il entre la taille du projet en KLSL (Kilos Lignes Source Livrés) et il choisit dans la deuxième case la méthode utilisée "Simple" ou "Intermédiaire" et dans la troisième case il choisit le mode utilisée "Organique", "Semi détaché" ou "Embarqué".

Si le chef du projet choisit la méthode "Simple" la fenêtre suivante s'affiche:Figure5-8



FigureV-9:Interface COCOMO simple

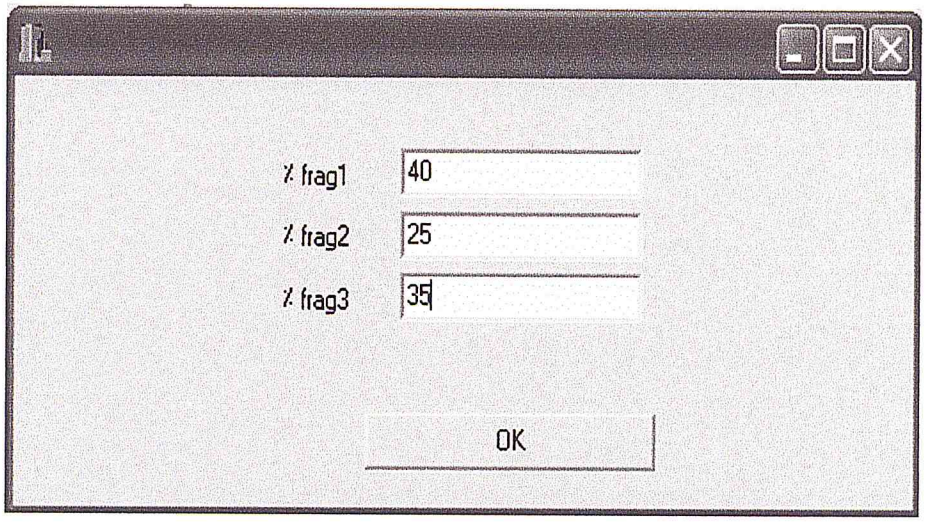
Et s'il choisit la méthode "Intermédiaire" elle s'affiche la fenêtre suivante:



FigureV-10: Interface COCOMO intermédiaire

À gauche de cette fenêtre"Figure5-9" on a les facteurs qui agissent sur les facteurs d'évaluation tel que chaque facteur possède cinq valeurs (très bas, bas, moyen, élevé, très élevé) et au chef de projet de sélectionner une et s'il ne sélectionne pas elle va se considérer comme une valeur moyenne, et les résultats des facteurs d'évaluation sont faits selon la méthode "COCOMO intermédiaire" indiquée dans le chapitre2.

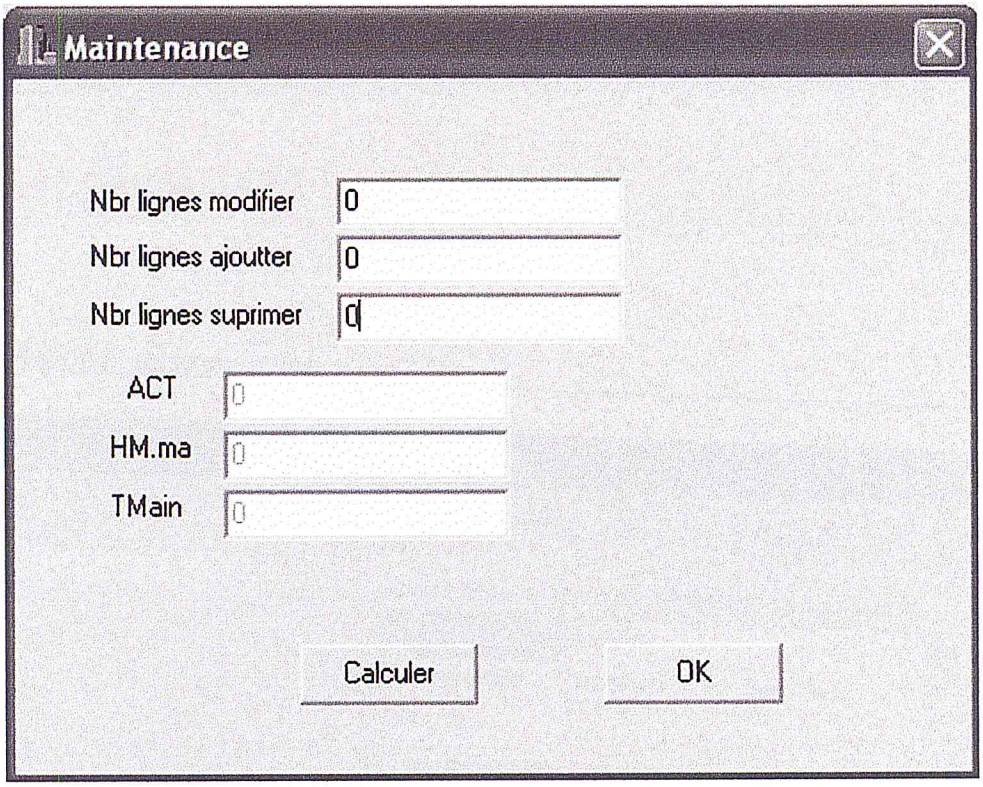
Après le calcule des facteurs d'évaluation de tout le projet, elle s'affiche la fenêtre dont la figure5-10 qui permet au chef de projet de faire entrer le pourcentage de temps et de coût qui va prendre chaque fragment.



FigureV-11: pourcentage des facteur d'évaluation pour chaque fragment

Avant d'établir le pourcentage si le chef du projet décide de faire une maintenance et de rétablir les facteurs d'évaluations la fenêtre "Maintenance" s'affiche. Figure5-11

Il suffit qu'il entre le nombre de lignes modifiés, ajoutées et supprimées et qu'il clique sur le bouton "Valider" pour calculer le nouveau HM "HM-ma" et le nouveau temps de développement "T-ma". La méthode de calcule est aussi indiquer au chapitre2.



FigureV-12: maintenance

Après l'établissement des facteurs d'évaluations le chef de projet revient à la fenêtre principale pour cliquer sur le bouton "Espaces de travail" pour consulter l'architecture des espaces de travail ou va s'exécuter chaque fragment "phase".

Pour cela elle s'affiche la fenetre dont la figure5-12, dans cette fenêtre on a pour chaque phase un tableau qui represente toutes les activités de la phase et ses ressources necessaires a leur exécution, un tableau ou on a la liste des ressources fournis pour réaliser chaque phase, on trouve aussi les valeurs des facteurs d'evaluations "temp et cout" pour chaque fragment "phase".

String nom activité : désigne le nom de l'activité.
String code activité : désigne le code de l'activité.
String numéro fragment: ce champ désigne le numéro de fragment qu'elle appartient cette activité.
Ressources *res: ce champ est considéré comme le pointeur qui pointe sur la liste des ressources nécessaire pour chaque activité.
Activité * act: c'est le pointeur qui pointe vers l'activité suivante.

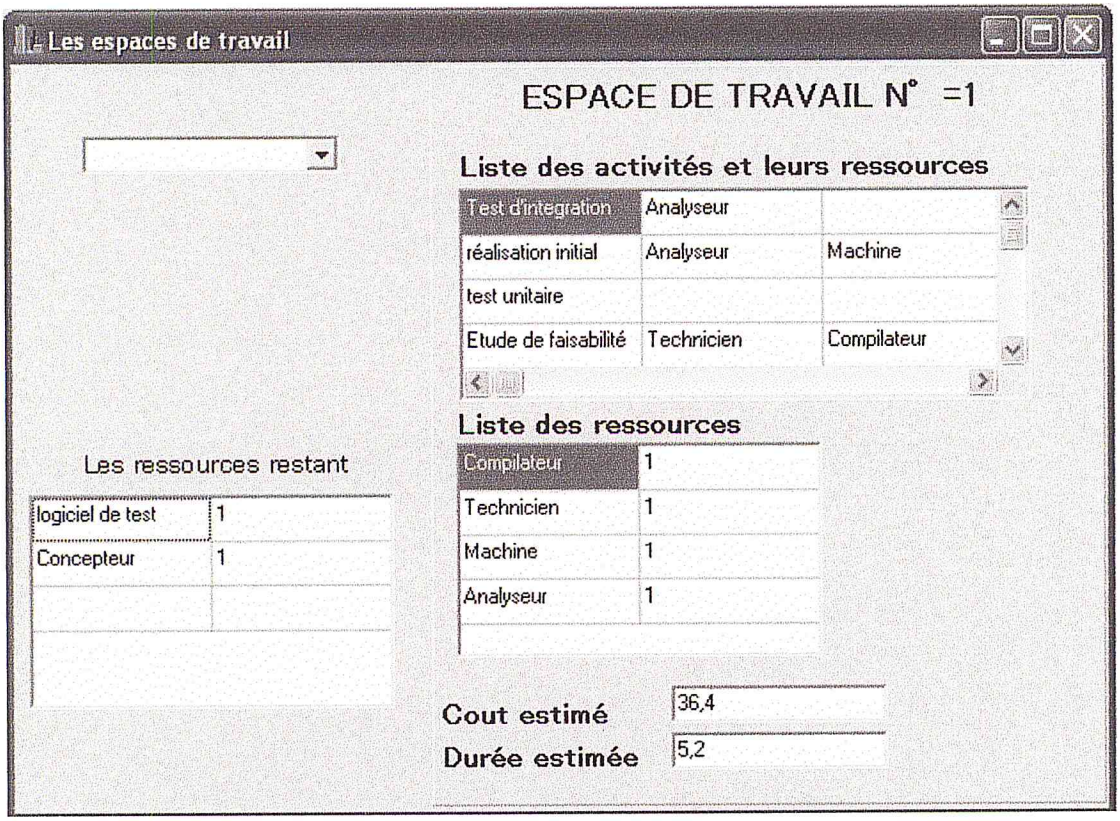
On réalité on a quatre listes d'activités (LISTEACT1, LISTEACT2, LISTEACT3, LISTEACT4) tel que chaque liste d'activité représente les activités d'un fragment.

Les ressources sélectionnées pour les affecter à une activité seront enchaînés dans une liste "Liste des ressources" dont la structure est la suivante:

```
Class ressources
{
Public:
String  nom ;
String code;
Bool état ;
Ressources *res;
} ;
```

String nom: pour désigner le nom de la ressource.
String code: pour désigner le code de la ressource.
Bool état: représente l'état de la ressource (Libre ou occuper). Ressources *res: c'est un pointeur pour la prochaine ressource.

Après la construction de chaque liste des ressources le pointeur "Ressources *res" de la liste des activités pointe immédiatement sur la tête de la liste des ressources construite.



FigureV-13: les espace de travails

V-2-c- Stocker le modèle de procédés dans la base de connaissance

En effet le stockage des connaissances se fait au fur et à mesure de la modélisation, tel que a chaque fois qu'on choisit une activité ou une ressource pour une activité ces derniers seront enregistrés et enchaîner dans une liste qui leur convient.

À chaque fois qu'on sélectionne une activité sera enchaînée dans une liste dont la structure est la suivante:

```
Class Activité
 {
 Public:
  String nom activité ;
  String code activité ;
  String numéro fragment;
  Ressources *res;
  Activité * act;
};
```

Conclusion

L'application a été réalisée dont le but d'offrir un outil d'aide pour la modélisation des procédés logiciels et on a essayé de simplifier les étapes de la modélisation par les remarquables richesses et fonctionnalités des composants visuels proposés par l'environnement de développement utilisé.

Conclusion générale

[1]: N.R.Jennings 1,T . J . Jhonson ,T.J.Normon , P.Faratin,P.O'Brien and B. Odgers<<agent autonome pour un gestionnaire de procédés busness>> 1Dept.electronic Engeniring, Queen mary § westfield College, Uniersity of London, London E1 4NS ,UK.2000.

[2]: Iiham Alloi , Sorana Cimpan , Flavio Oquendo <<une structure logiciel pour la modélisation des procédés :support des décision et des contrôle >>

[3]: M. stambouli Ahmed , M.Tassis Ahmed <<Interrogation d'une base de données réparties et hétérogènes a l'aide d'agents mobile>> Thèse de d'ingéniorat 2004.

[4]: F.Aussat << Modélisation et exécution des procédés logiciels à base d'agents intelligents>> thèse de magister ; USTHB ; 2003.

[5]: Nadinne Richard<<Description des Comportements d'agents autonomes évoluant dans des mondes virtuels>> thèse de doctorat, école nationale supérieure de télécommunications,2001.

[6]: Imed Jarras et Brahim chaib-draa <<Aperçu sur les systèmes muli-agents>>2002.

[7]: Marc –André Labri <<langage de communication agent basé sur les engagements par l'entremise des jeux des dialogues.

Conclusion générale

La complexité croissante des logiciels insuffisamment prise en charge par les méthodes traditionnelles et centralisée de développement, conjuguée à l'évolution tout aussi croissante de la technologie, ont contribué à l'arrivé de nouveaux modes et méthode de développement basé sur l'utilisation optimal des ressources humains et matériels distribué.

Parmi ces modes, l'utilisation d'agents intelligents pour l'ingénierie des logiciels présente un intérêt certain par ses capacités d'autonomie' d'action et d'interaction entre les espaces de travail

Dans notre thèse nous définissons une nouvelle approche pour la modélisation et l'exécution des modèles de procédés logiciels ; une structure générale du modèle de procédé logiciels système multi agents, tel que notre travail se focalise sur la partie supérieur du système "Agent Superviseur", cependant les autre parties restent à définir.

Dans ce mémoire notre agent a été réalisé dans le but à offrir un outil d'aide pour la modélisation des procédés logiciels et on a essayé d'aller plus loin dans l'implémentation de cet agent mais vue à ses nombreuses fonctions et à la complexité de la réalisation de ces fonction on a arrivé juste a faire un pas vers le bon chemin de la modélisation des procédé logiciels à base d'agent intelligent.

Références

[8] : Wooldridge, M. et Jennings, N, R. (1995).
<< Intelligent agents: Theory and practice>>
Knowledge engineering review, 10(2).

---

[9]: Jacques Ferber : loforia, université de pierre et marie curie,
<<les systèmes muli-agents : vers une intelligences collectives>>
Inter Edition 1997.

---

[10]: Leblanc Gerad
<< C++ builder>>

---

[11]: Pierre –Alain Muller
<<Modélisation objet avec UML>>
EYROLLES ,2001.

---

[12]: Adina Magda Florea professeur à l'université « politechnica » de Bucarest<<
Agents et Systèmes multi_agents>>
http ://truring.cs.pub. ro/auf2/

---

[13] : Grégor joerie ,Christoph Klauck,Ottehein herzog
<<gestionnaire de procédé distribué et dynamique basé technologie d'agents >>
Proceeding of the sixth European software engineering conference (ESEC/FSE97).

---

[14]: A. Mostfai
<<La fédération dans les environnements centrés procédés logiciels >>
Thèse de magister ; USTHB ; 2002.

---

[15]: Romaric CHARTON.<<Des agent intelligents dans un environnement de communication multimédia :vers la conception des services adaptatifs>>
inter Edition 1998.

---

[16] : Peiwei Mi et walt Scacchi
<< Processus d'intégration dans les environnements CASE>>
Dept.of Computer and information science,
University of southren califonia;
Los Angeles.
HTTp://citesseer.nj.nec.com

---

[17]: Chantal morley-jean Hugues - Bermard leblanc)
<<UML pour l'analyse d'un système d'information>> Edition2
Institut National de télécommunication d'évry.

---

[18]: AKLOUF Y.§DRIAS H
<< une architecture de place de marché à base d'agents mobiles et intelligents>>, in porc of ISPS '03,alger,mai 2003.

---

[19]: M.bouziane Mohamed et M. chiheb djamel.
<<Élaboration d'un système a base de connaissance pour les opérateurs économique algériens>>
Thèse d'ingénieras 2003/2004 université de Blida.

---

[20]: Alf Inge Wang
<<Utilisant un environnement basé agent pour supporter les procédés logiciels coopératifs>> thèse de doctorat.
Université norvégienne des sciences et technologie ;
Dept.of Computer and information science,
NTNU, trandheim, Norway, february 5th 2001.
http://idi.ntnu.no/grupper/su/publ/alfw/seke2002.aiw.pdf

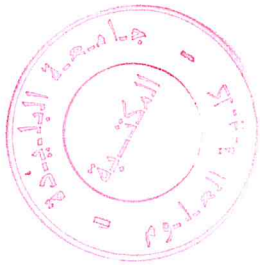
---

[21]: Alf Inge Wang ,Reider Conradi et chunnian liu :
<< Architecture multi –agents pour l'ingénierie logiciel coopérative>>
In porc. of the eleventh international conference on soufware ingineering and knoledge engineering ? seke 99 page 1-22,Kaiserslautern,germany,17-19 june 1999.

---

[22]: Ian Sommerville
<<le génie logiciel et ses applications>>
Université de Strathclyde, Royaume-uni

LISTE DES FIGURES :



Liste des tables:

Annexes

Les plateformes de génération d'agents

Il existe un nombre important d'environnements de développement des applications orientées agents : il y a aussi bien des produits commerciaux que des logiciels dans le domaine public.

Parmi les plates-formes fournies comme logiciels libres, il y a quelques plates-formes plus connues pour avoir été utilisées dans le développement de plusieurs applications : MACE, ZEUS, et MADKIT pour les agents cognitifs, et SWORM pour les agents réactifs. Il faut noter que cette liste n'est pas unique, et qu'il y a aussi d'autres plates-formes qui ont été utilisées avec beaucoup de succès pour bâtir diverses applications.

MACE (Gasser e.a., 1987) est le premier environnement de conception et d'expérimentation de différentes architectures d'agents dans divers domaines d'application. Dans MACE, un agent est un objet actif qui communique par envoi de messages. Les agents existent dans un environnement qui regroupe tous les autres agents et toutes les autres entités du système. Un agent peut effectuer trois types d'actions : changer son état interne, envoyer des messages aux autres agents et envoyer des requêtes au noyau MACE pour contrôler les événements internes. Chaque agent est doté d'un moteur qui représente la partie active de l'agent. Ce moteur détermine l'activité de l'agent et la façon dont les messages sont interprétés. MACE a été utilisé pour développer des simulations d'applications distribuées.

ZEUS (Nwama e.a., 1999) est une plate-forme multi agents conçue et réalisée par British Telecom (Agent Research Programme of BT Intelligent Research Laboratory) pour développer des applications collaboratives. ZEUS est écrit dans le langage Java et il est fondé sur les travaux de la FIPA. L'architecture des agents ZEUS est similaire à la majorité des agents collaboratifs. Elle regroupe principalement les composantes suivantes:

Une boîte aux lettres et un gestionnaire de messages qui analyse les messages de la boîte aux lettres et les transmet aux composantes appropriées.

Un moteur de coordination.

Un planificateur qui planifie les tâches de l'agent en fonction des décisions du moteur de coordination, des ressources disponibles et des spécifications des tâches.

Plusieurs bases de données représentant les plans connus par l'agent, les ressources et l'ontologie utilisée.

Un contrôleur d'exécution qui gère l'horloge locale de l'agent et les tâches actives.

L'environnement comporte trois bibliothèques : une avec des agents utilitaires, une avec des outils pour la construction des agents, et une avec des composants agents.

ZEUS met un fort accent sur la méthodologie de développement, fondée sur la notion de rôle. ZEUS a été utilisé pour développer plusieurs applications réelles comme les ventes aux enchères et la simulation de la fabrication des ordinateurs. Les caractéristiques des domaines d'applications de ZEUS ont été définies par les concepteurs ; parmi ces caractéristiques, on peut mentionner :

Chaque agent crée un plan qui nécessite un raisonnement explicite pour atteindre son but ; la résolution de problèmes nécessite une coopération entre agents ; le rôle de chaque agent consiste à contrôler un système externe qui réalise une tâche du domaine d'application, la résolution de problème est ainsi effectuée par ce système externe et contrôlée par les agents.

MADKIT (Madkit, 2003) est une plate-forme développée par le Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM) de l'Université Montpellier II. MADKIT est libre pour l'utilisation dans l'éducation. MADKIT est écrit en Java et est fondé sur le modèle organisationnel Alaadin. Il utilise un moteur d'exécution où chaque agent est construit en partant d'un micronoyau. Chaque agent a un rôle et peut appartenir à un groupe. Il y a un environnement de développement graphique qui permet facilement la construction des applications.

SWARM (Minar e.a., 1996) est une plate-forme multi agents avec agents réactifs. L'inspiration du modèle d'agent utilisé vient de la vie artificielle. SWARM est l'outil privilégié de la communauté américaine et des chercheurs en vie

artificielle. L'environnement offre un ensemble de bibliothèques qui permettent l'implémentation des systèmes multi agents avec un grand nombre d'agents simples qui interagissent dans le même environnement.

De nombreuses applications ont été développées à partir de SWARM qui existe aujourd'hui implémenté en plusieurs langages (Java, Objective C).

Voici un tableau (tableau1) qui représente un ensemble de plateformes "projets universitaires" ainsi l'organisation qui les créer et leur langage de base:

Produit	Organisme	Langage	Description
Produit	Organisme	Langage	Description
Agent Building Shell (ABS)	University of Toronto	COOrdination Language (COOL)	Architecture d'Agents
Agent TCL (D'Agents)	Dartmouth University	Tcl	Agents mobiles
Architecture type-based Development Environment (ADE)	University of Potsdam Dept. of Computer Science Professorship of Software Engineering TAXT	Java	Environnement de développement
Bee-gent	Toshiba Corporation Systems and Software Research Laboratories	Java	Environnement de développement
Bond Distributed Object System	Purdue University	Java	Environnement de développement
Cable	Logica Corporation	Agent Definition Language, C++	Architecture de système
Cybele	Intelligent Automation, Inc.	unknown	Infrastructure logicielle
DECAF Agent Framework	University of Delaware	Java	Environnement de développement
dMARS	Australian Artificial Intelligence Institute Ltd.	C, C++	Environnement d'implantation et de développement d'Agents
EXCALIBUR	Technical University of Berlin, German National Research Center for		Architecture d'Agents autonomes

Annexe1: les plateforme de génération d'agents

	Information Technology, Research Institute for Computer Architecture and Software Technology		
FarGo	Distributed Systems Group, Technion - Israel Institute of Technology	Java	Environnement de développement
Gypsy	Technical University of Vienna	Java	Agents mobiles
Hive	The Media Lab Massachusetts Institute of Technology	Java	Kit de développement d'Agents
Infospiders	University of California San Diego - Computer Science Dept.		Agents adaptatifs d'information
JAFMAS	University of Cincinnati	Java	Environnement de développement
JATLite	Stanford University	Java	Packages Java pour Multi-Agents
JATLiteBean	University of Otago	Java	Composants JavaBean
JIAC	Technische Universitat Berlin	Java	Architecture d'Agents
Kasbah	Massachusetts Institute of Technology		Agents pour le commerce électronique
KLAIM	Universita' Di Firenze	Klaim	Agents mobiles
Knowbot® System Software	CNRI	Python	Agents mobiles
LALO	CRIM	LALO	Environnement de développement
Mobiware Middleware Toolkit	Columbia University	Java	Environnement de développement
MOLE	University of Stuttgart	Java	Agents mobiles
Multi-Agent Modeling Language (MAML)	Central European University	MAML	Langage de programmation
MultiAgent Systems Tool (MAST)	Technical University of Madrid	C++	Agents hétérogènes
Open Agent Architecture	SRI International		Environnement de développement
ProcessLink	Stanford University		Environnement de développement
RETSINA	Carnegie Mellon University		Agents communiquants
Social	DFKI (German Research	Java	Environnement de

Interaction Framework (SIF)	Institute for AI)		développement
Sodabot	MIT Artificial Intelligence Lab		Environnement de développement
TuCSoN	Universita di Bologna		Environnement de développement

Tableau1: Ensemble de plateforme dégénération d'agent

I- Les Aglets

En 1996 Big Blue (IBM) met à disposition " Aglets Worbench " qui regroupe une librairie sous forme d'un package Java, ainsi qu'un serveur d'aglets (Tahiti) . " Aglets Workbench " sera renommé en 1998, ASDK (Aglets Software Developement Kit). La ASDK est destiné à faciliter la création d'Agent intelligent et mobile.

Aglets vient des mots Agent et Applet. Quand nous parlons de serveur d'Aglets, ceci est équivalent à un serveur d'Agent pour ce système.

L'API Aglet Java est une norme proposée pour développer des Agents mobiles basé sur java. L'API a été développé par une équipe de chercheurs du laboratoire de recherche d'IBM Tokyo au Japon; son but est de fournir une plate-forme uniforme pour les Agents mobiles dans un environnement hétérogène tel que celui de l'Internet.

L'API permet d'écrire en une seule fois les applications, qui s'exécuteront partout avec les avantages liés à Java. C'est-à-dire, que les Aglets fonctionneront sur chaque machine qui supporte l'API. On n'aura plus à se soucier de la couche matériel ou du système d'exploitation, ni de la manière dont le code de l'API a été programmé.
Dans ce projet, les Aglets ont été compilés et lancés seulement sur des stations Sun SOLARIS.

Voici un diagramme qui décrit l'environnement des aglets

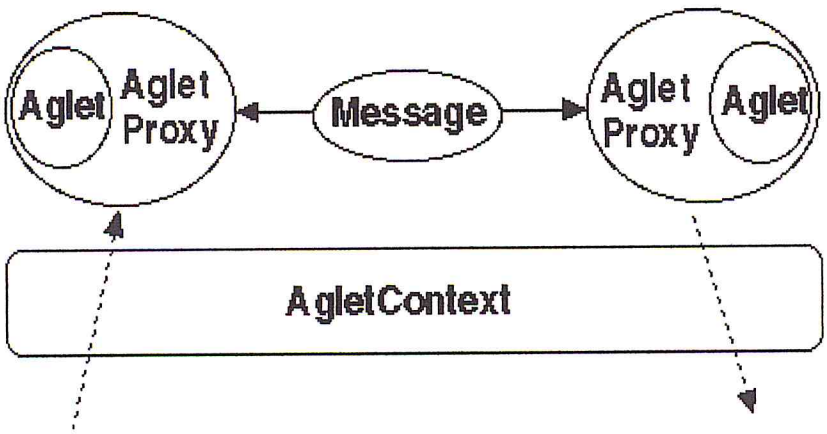


Figure1

Le modèle d'objet d'Aglet définit un ensemble de techniques permettant de créer des Agent mobiles dans un réseau du type étendu tel que le réseau Internet. Les principales classes trouvées dans ce modèle sont Aglet, contexte, procuration, message, itinéraire, et identifier

- Un Aglet est un objet mobile de Java qui visite les serveurs où les Agents sont autorisés, dans un réseau informatique. Il est autonome, et réactif.
- Un contexte est un objet qui fournit des moyens pour mettre à jour et contrôler des aglets dans un environnement uniforme d'exécution où le système hôte est immunisé contre des aglets malveillants. Un noeud dans un réseau informatique peut accueillir des contextes multiples.
- Un proxy est un représentant d'un aglet. Il sert de bouclier à l'aglet pour le protéger contre l'accès direct à ses méthodes publiques. Le proxy fournit également la transparence à l'emplacement pour l'aglet. C'est-à-dire qu'il peut cacher le vrai emplacement de l'aglet.
- Un message est un objet échangé entre aglets. Il permet d'échanger des messages synchrones aussi bien que des messages asynchrones entre les aglets. La communication par messages peut être employée par des aglets pour collaborer et échanger l'information.
- Un message manager permet de gérer la l'arrivée des messages.
- Un itinary est un plan de trajet pour les aglets. Il fournit une manière commode pour fournir un routage des aglets.
- Un identifier est lié à chaque aglet. Cet identificateur est globalement unique et immuable durant toute la vie des aglet.

II- Les méthodes de la classe aglet

Les méthodes supportées dans la classe aglet permettent la création, clonage, dispatching, retrait, désactivation, lancement, disposition, et la messagerie:

- La création d'un aglet a lieu dans un contexte. Le nouvel aglet est assigné un identificateur, inséré dans le contexte, et initialisé. L'aglet commence à exécuter dès qu'il sera avec succès initialisé.
- Le clonage d'un aglet produit une copie presque identique de l'aglet initial dans le même contexte. La seule différence est que les aglets sont relancés.
- Dispatcher un aglet d'un contexte vers un autre le retirera de son contexte actuel et l'insérera dans le contexte du destinataire, où son exécution sera relancée.
- Le retrait d'un aglet le retirera de son contexte actuel et l'insérera dans le contexte où le retrait a été demandé.
- La désactivation d'un aglet permet de l'enlever temporairement de son contexte actuel et de l'enregistrer dans la mémoire secondaire. L'activation de l'aglet va le restaurer en mémoire secondaire.
- Disposer un aglet le stoppera dans son exécution actuelle et la retirera de son contexte actuel.
- La messagerie permet l'envoie, la réception, et la manipulation des messages de manière synchrone et asynchrone.

Ces fonctions sont schématisées dans ce diagramme Figure2

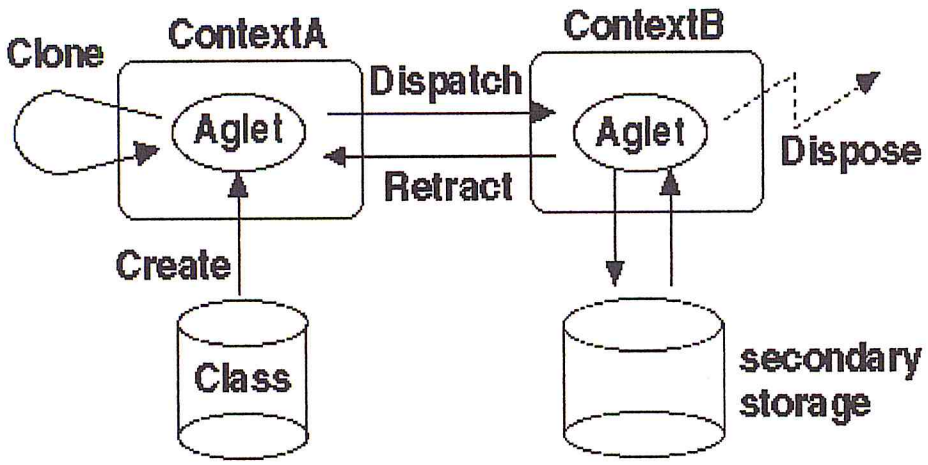


Figure2

III- Cycle de vie d'une Aglet

Il existe trois types d'opérations fondamentales dans le cycle de vie d'une Aglet

- Create : une Aglet est créée relativement à un aglet context. Après sa création, une Aglet est initialisée et son thread d'exécution commence.
- Clone : cette opération produit une nouvelle Aglet qui possède une nouvel identifiant. Cependant, le thread d'exécution est réinitialisé.
- Dispose : lorsque le travail de l'aglet est terminé, on peut en disposer. Ce processus termine l'exécution est fait appel au garbage collector.

Durant la vie de l'Aglet, plusieurs choses peuvent être faite :

- Dispatch : une Aglet est sortie de son contexte et envoyé vers un autre, généralement un contexte distant où sa méthode run() recommence.
- Retract : l'Aglet est enlevée de son contexte distant afin d'être replacée dans son contexte local.
- Desactivate : l'exécution est stoppée mais l'Aglet reste dans son contexte.
- Activate : le contraire.

Les Aglets ne peuvent exister qu'au sein d'un contexte. Ce contexte est en fait un conteneur qui permet d'utiliser les Aglets dans n'importe quelle autre application Java.

En ce qui concerne la programmation, le contexte d'une Aglet procure plusieurs services dont la création d'une Aglet. Dans l'exemple suivant le contexte est appelé cxt et va nous permettre de créer une nouvelle Aglet :

```
AgletRuntime runtime = AgletRuntime.init(null);
cxt = runtime.createAgletContext("agletcontext");

AgletProxy proxy = cxt.createAglet(null, "AnAglet", null);
```

De la même manière, une Aglet peut à son tour créer une autre Aglet en se servant de la méthode getAgletContext() :

```
AgletContext ac = getAgletContext();

URL codebase = new URL("file://c:/agletcentral");

AgletProxy proxy = ac.createAglet(codebase, "AnAglet", null);
```

Avec comme arguments :

1. codebase -> chemin du fichier source
2. "AnAglet" -> nom de l'Aglet
3. Le 3ème argument représente les paramètres passés à la méthode onCreation()

Le serveur TAHITI (Figure3) permet entre autre de créer et charger les Aglets, de les dispatcher vers un nouveau contexte et de les en retirer. Lors de la création d'une Aglet, les opérations suivantes sont effectuées :

- Charger le fichier class
- Instancier l'Aglet
- Etablir l'Aglet dans son contexte
- Invoquer la méthode onCreation()
- Convoquer la méthode run()

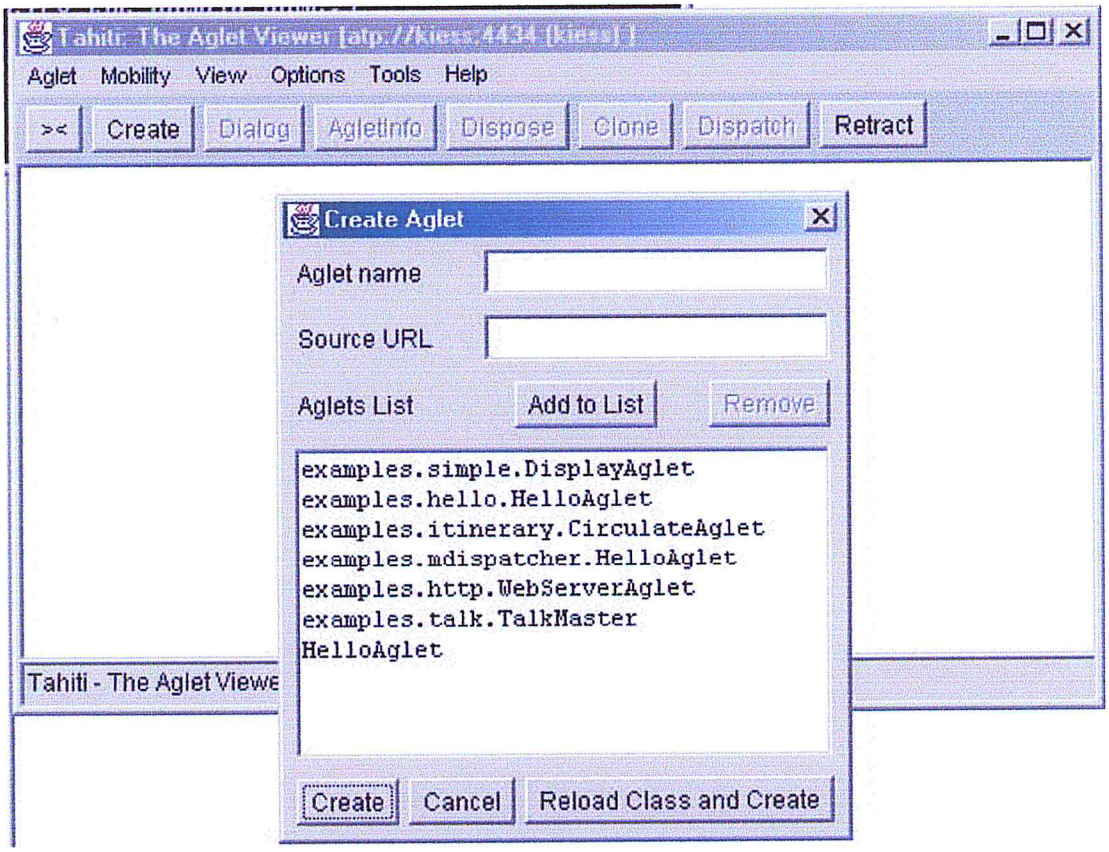

Aglet Mobility View Options Tools Help
>< Create Dialog AgletInfo Dispose Clone Dispatch Retract
Create Aglet
Aglet name
Source URL
Aglets List
Add to List
Remove
examples.simple.DisplayAglet
examples.hello.HelloAglet
examples.itinerary.CirculateAglet
examples.mdispatcher.HelloAglet
examples.http.WebServerAglet
examples.talk.TalkMaster
HelloAglet
Create
Cancel
Reload Class and Create

Figure3

CHAPITRE III

Architecture interne de l'agent superviseur